

QUANTIFYING RELATIONSHIPS BETWEEN STORAGE AND DISCHARGE IN A
LARGE DATABASE OF REFERENCE WATERSHEDS ACROSS THE CONTINENTAL
US

by

Holly Guest

A thesis submitted to Johns Hopkins University in conformity with the requirements for the
degree of Master of Science.

Baltimore, Maryland

April 2015

Abstract

So far there is not a globally known system for catchment classification that will aid in improved models and communication amongst hydrologists. This paper combines two common methods used to analyze the hydrologic response of watersheds, one being storm-flow hydrograph separation and the other being the store and release of water as baseflow, to improve our ability to develop such a classification. The two distinctions are combined using a catchment sensitivity function that was developed by James Kirchner (2009) and an event flow response model that is derived under the assumptions following the US Soil Conservation Service-Curve Number (SCS-CN) method. The model is applied to 671 small-to-medium sized watersheds with minimal human impact across the United States to extract parameters representing the relationship between catchment storage and event runoff. Parameters from the model that are comparable across watersheds are mapped spatially to locate trends to better understand how basins act based off of location. The results show particular trends along the Appalachian Mountains where the range of storage values in the Blue Ridge Mountains are collectively larger than the watersheds in the Ridge and Valley Appalachians that collectively show smaller range of storage values. There are also noticeable spatial clusters of the model's parameters, such as the maximum storage parameter showing low values clustered around Iowa, Illinois, Missouri, and Arkansas. Ciaran Harman is the advisor/reader for this paper.

Table of Contents

1	Introduction	1
2	SCS-CN Method.....	4
3	Revising the SCS-CN Method.....	8
4	Storage-Discharge Relationship Theory.....	12
5	Details of Analysis	16
5.1	Data Description	16
5.2	Sensitivity function Regression Analysis.....	17
5.3	Defining Storage	21
5.4	Storm Flow Regression Analysis.....	21
6	Results	24
7	Discussion	33
8	Conclusion.....	35
9	References.....	36
10	Appendix A: Flow Charts	38
11	Appendix B: Code	40
12	Curriculum Vitae.....	58

Table of Figures

Figure 1 Hyetograph and Hydrograph for SCS-CN Method.....	6
Figure 2 Hyetograph.....	9
Figure 3 Hydrograph	9
Figure 4 Bins evenly distributed.....	18
Figure 5 Bins with satisfactory standard error	19
Figure 6 Storage-Discharge Relationship.....	20
Figure 7 An example of observed and predicted runoff for a watershed	24
Figure 8 NSE values of each basin for the sensitivity function	26
Figure 9 NSE values of each basin for the storm flow function	26
Figure 10 Storage-Discharge relationship with no precipitation and minimal evapotranspiration for Basin 12013500.....	27
Figure 11 predicted storm flow for Basin 12013500	27
Figure 12 Storage-Discharge relationship with no precipitation and including evapotranspiration for Basin 6917000	27
Figure 13 predicted storm flow for Basin 6917000.....	27
Figure 14 NSE values mapped spatially for the sensitivity function for every basin that met the analysis criteria	28
Figure 15 NSE values mapped spatially for the storm flow final analysis.....	28
Figure 16 Exponent average parameter for the sensitivity function.....	29
Figure 17 Exponent curvature for the sensitivity function	29
Figure 18 Maximum storage value for each basin mapped spatially	30
Figure 19 Minimum storage value for each basin mapped spatially	30
Figure 20 Total storage for each basin mapped spatially	31
Figure 21 Vrange values along the Appalachian Mountains.....	31
Figure 22 bcurvature values along the Appalachian Mountains.....	32
Figure 23 Plot of aridity index and Vrange to see any correlation.....	32

1 Introduction

What makes catchments different from each other? How do we go about classifying them based off of their differences? These questions exist in part because there is yet to be a classification system for catchments in the field of hydrology that will yield consistent answers to these questions. There is no catchment classification that is synonymous to say the Reynolds number that classifies water flows in a fundamental way (i.e. laminar vs turbulent) (Wagener, Sivapalan, Troch, & Woods, 2007). An agreed upon classification system would have many benefits such as improving global communication around hydrology or distinguishing a poorly understood hydrologic model from a well understood one (Wagener et al., 2007). A classification system would also benefit the engineer or researcher that endeavors to model catchments for civil design and/or conservation. This paper focuses on two features of catchment hydrologic dynamics that distinguish one catchment from another. The first is how catchments respond to storm flow, and the second is how catchments store and release water during baseflow. These have typically been studied separately, but here we present an approach to analyzing these features in a unified way. Storm flow and the releasing/storing of water in a watershed are critical factors that distinguish catchments, so combining the two models in a unified framework will help locate spatial patterns that will aid catchment classification. The unified model will hopefully lead to insight and more understanding in how to approach catchment classification. There is obviously a large complexity in a catchment with many variables, but understanding these variables through data could lead to such a classification system (Wagener et al., 2007). This approach of observing patterns across a landscape of catchments and then hypothesizing reasons to the patterns is akin to the Darwinian approach first introduced by Charles Darwin (Harman & Troch, 2014). It connects past data with the

present day state of catchments (Harman & Troch, 2014). So how is this past data represented or summarized to give inference to the present day state? How is it classified to distinguish catchments? It has been suggested that the distinguishing functions of the classification system should include partitioning (dividing water into different flowpaths), storage (water stored in different areas of the catchment), and the release of water (related to discharge, evapotranspiration, etc.) (Wagener et al., 2007). This analysis is an effort to understand or to identify potential spatial relationships and patterns in catchment hydrology that will lead to such a classification system.

The model used to analyze the storm flow response is derived from is the Soil Conservation Service (SCS) Curve Number method (SCS-CN). The SCS, now called the National Resources Conservation Service (NRCS), was established in 1933 to oversee conservation projects related to soil (Woodward, Hawkins, Hjelmfelt, Mullem, & Quan, 2002). One of their beginning tasks was to develop a Hydrology Guide to aid engineers in watershed planning (Plummer & Woodward, 1998). In 1949 L.K. Sherman suggested the development of empirical runoff models derived from plots of runoff versus rainfall (Woodward et al., 2002). Victor Mockus in 1949 proposed that the runoff could be estimated from soil type, land use, antecedent conditions, storm duration and magnitude, and the average annual temperature during the storm event (Mishra & Singh, 2003). The SCS Hydrology Guide simplified his equation in 1954 and later in 1964 Mockus produced runoff curve numbers based off of watershed data to produce the now well known SCS-CN methodology (Woodward et al., 2002).

The SCS-CN methodology uses a water balance equation and a proportionality rule to develop a relationship between the initial storage (or “abstraction”) of precipitation, maximum potential storage, and storm event discharge. In typical applications of this method, the initial and potential storage are determined through a curve number that is selected based off of the watershed’s soil type, land use, hydrologic conditions, and antecedent moisture conditions. This curve number and a parameter within the analysis known as initial abstraction, or the amount of water that enters storage before runoff occurs, are empirically determined and sensitive to errors (Boughton, 1989). The initial abstraction and storage relationship are hard to determine and provide small confidence in the analysis. However, the derived proportionality rule and the water balance equation from the SCS-CN method are used for this study as the storm flow model. Instead of using tabulated CN values to predict hydrographs, we will use data from a large number of storms in a large number of watersheds to examine the validity of this rule, and between-catchment variations in the relationship it reveals.

This is where the second model describing the storage and release of water comes into play. Storage must be defined now that it is no longer determined through the CN. The model here will use a baseflow storage-discharge model to empirically quantify relative storage values to be used in place of the storage values calculated with the CN. This idea of examining the relationship between storage and baseflow discharge from empirical analysis of the hydrograph recession has been developing since Brutsaert and Nieber (1977) modeled drought flow or baseflow by plotting discharge (Q) against the rate of change of discharge ($-dQ/dt$) and interpreting the relationship through the Dupuit-Boussinesq aquifer model (Brutsaert & Nieber, 1977). Wittenberg and Sivapalan (1999) also incorporated the idea of a nonlinear reservoir to

see how the storage-discharge relationship changes seasonally by including evapotranspiration losses in their baseflow separation (Wittenberg & Sivapalan, 1999). Kirchner developed a way to represent the sensitivity of discharge to storage as a function of discharge using data when precipitation and evapotranspiration are negligible (Kirchner, 2009). Kirchner's model is adapted for this analysis with a few minor changes in how the storage-discharge relationship is estimated mathematically.

So, does describing storm flow this way help in coming up with a classification system? What can be said from the past about how basins react that will help in catchment characterization? In an effort to answer these questions the analysis is applied to 671 small watersheds across the continental US with minimal human impact to see patterns arise (Newman et al., 2014). The storage values and model parameters are plotted spatially to see trends or patterns across the landscape that will aid in understanding how to develop a catchment classification system.

2 SCS-CN Method

The SCS-CN method separates an event rainfall hyetograph into actual infiltration (F), total rainfall (P), and initial abstraction (I_a). Initial abstraction is the amount of infiltrated water before runoff is generated. Assuming the Hortonian model, this parameter would account for the initial infiltration, when the infiltration capacity of the soil is greater than the rainfall intensity. Actual infiltration is the amount of infiltrated water that accumulates once runoff is generated, or all of the infiltrated water from the event excluding I_a . Total rainfall is the amount of rain accumulated throughout a storm event. Mishra and Singh (2003) review interpretations and controls on the SCS-CN parameters. The magnitude of the initial abstraction depends on

its interception, surface storage, and infiltration. It can also vary with evapotranspiration prior to the event. If evapotranspiration is high, then it is likely that initial abstraction will be large as well. Many climatic factors such as radiation, albedo, humidity, and temperature affect evapotranspiration, and so also play a key role in initial abstraction. Initial abstraction also affects the magnitude of the runoff (Q). If the initial abstraction is large, then more of the precipitation will have to go into initial abstraction before runoff can begin, thus reducing runoff. The SCS-CN method also introduces a potential maximum retention term, S . This term is defined as the maximum storage of a watershed, or the maximum amount of water that a watershed can retain. This value is what the curve-number approximates. The curve number (CN) value depends on soil type, land use, hydrologic conditions, antecedent moisture conditions, and consequently the climate (Dingman, 2002). The CN is a range of values from 0 to 100, but are typically found in a range from 40 to 98. Maximum retention, as defined according to the parameters, is expressed in Equation 1.

$$S = \frac{1000}{CN} - 10 \quad \text{Equation 1}$$

Figure 1 shows a hydrograph that separates the amount of direct surface runoff (Q) from the baseflow (Q_b). Within the hydrograph, the maximum potential runoff can be defined as the total rainfall (P) minus the initial abstraction (I_a), which gives the maximum amount of runoff that can occur given the magnitude of the rainfall. If all of the rain contributes to runoff then runoff is equal to the maximum potential runoff.

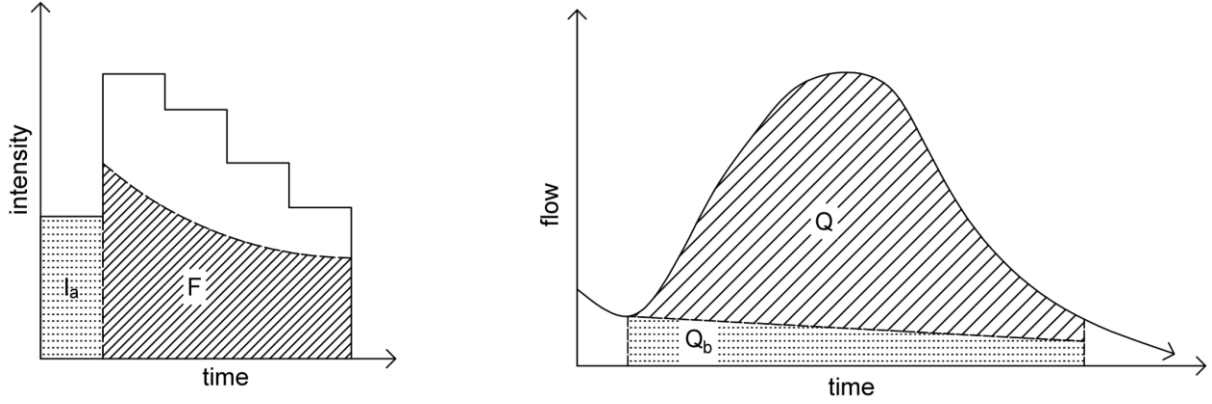


Figure 1 Hyetograph and Hydrograph for SCS-CN Method

The method is based on a mass balance equation (Equation 2) and two hypotheses.

$$P = I_a + F + Q \quad \text{Equation 2}$$

The first hypothesis is a proportionality assumption shown in Equation 3.

$$\frac{Q}{P - I_a} = \frac{F}{S} \quad \text{Equation 3}$$

This states that direct runoff (Q) divided by the maximum potential runoff ($P - I_a$) is proportional to infiltration (F) divided by the potential maximum retention (S). As Q approaches P , F approaches S . This equality is derived from Mockus's (1949) model expressed in Equation 4. His model assumes that the maximum potential runoff (P) increases linearly with time, or the cumulative rainfall grows linearly throughout the storm duration (Mishra & Singh, 2003). This allows for P to be defined as the mean intensity of the storm

times the time. It can also be derived from Horton's first-order storage hypothesis and the water balance equation.

$$\frac{F}{S} = \left(1 - e^{-P/S}\right) \quad \text{Equation 4}$$

This ratio also approximates the exponential as a first order Euler's continuous fraction within the derivation as shown in Equation 5 to remove the exponential.

$$\frac{F}{S} \approx \left(1 - \frac{1 - \frac{P}{S}}{1 - \frac{P}{S}}\right) = \left(1 - \frac{S}{S+P}\right) = \frac{P}{S+P} \quad \text{Equation 5}$$

Now, Equation 5 can be put into the water balance equation in Equation 2 with I_a equal to zero, and rearranged to obtain that the ratio of Q to P is equal to the ratio of F to S . Now the equality demonstrated in Equation 6 holds.

$$\frac{Q}{P} = \frac{P}{S+P} = \frac{F}{S} \quad \text{Equation 6}$$

The second hypothesis relates I_a to S according to Equation 7 where I_a varies according to λ and a CN value.

$$I_a = \lambda S \quad \text{Equation 7}$$

The λ value is typically set to 0.2, which is a value determined from rainfall-runoff records for only a select region of watersheds with areas less than 10 acres. This value holds bias according to the climate of the region and geologic conditions since its calibration was on a specific set of watersheds. Also, when plotted it produces large scatter; so the value of 0.2 is arbitrary even for this basin set and should not be held in confidence. Other studies have varied λ along the range of (0, 0.3), but still do not give a good agreement (Bosznay, 1989).

3 Revising the SCS-CN Method

The SCS-CN method is empirically construed, and does not accurately represent watersheds due especially to the poor characterization of the λ and CN values. However, the use of the water balance equation and the proportionality rule can still be used to represent a watershed's response to storms by redefining some terms from the SCS-CN method. The initial abstraction will no longer be based off of a reduction factor from S . Also, storage will not be modeled based on tabulated CN.

Figure 2 shows a schematic hyetograph of a rain event with a total amount, W (this is equivalent to P above). Initial abstraction, V_I is now defined as the storage that must be filled before any event runoff occurs; V_R is now defined as the increase in the volume of water retained in the catchment after initial abstraction and discharge are accounted (Dingman, 2002). The remaining water is what contributes to the storm runoff, Q_s , and the baseflow is Q_b . The baseflow is the water flow in the stream that is not part of the storm flow, and is assumed here to decrease exponentially with time over the event initializing from the stream flow before the storm event.

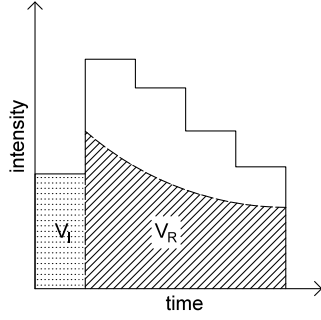


Figure 2 Hyetograph

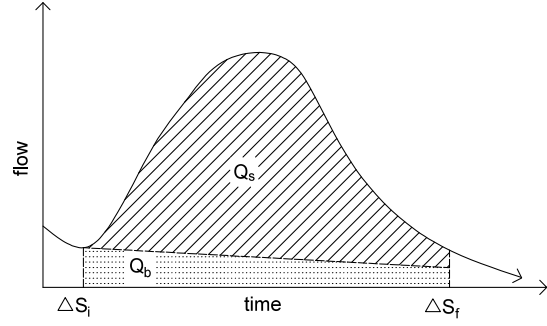


Figure 3 Hydrograph

Figure 3 shows an initial time, t_i where the event begins, and a final time, t_f at the end of the event. The catchment storage at these times is ΔS_i and ΔS_f respectively. For this analysis, V_I is described as,

$$V_I = \Delta S_{min} - \Delta S_i \quad \text{Equation 8}$$

where ΔS_{min} is the minimum storage capacity for the watershed. ΔS_{min} is defined relative to the mean storage, so ΔS_{min} is the storage that must be filled if ΔS_i is the mean storage. V_I equals ΔS_{min} if the initial catchment storage is the mean storage, but it will typically vary from storm to storm. Thus V_I can be negative if the catchment storage is already above ΔS_{min} .

The proportionality rule defined previously and restated in Equation 9 is the ratio of the rainfall infiltrated after initial abstraction (V_R) over the maximum rainfall that could be retained (V_{max}) to the surface runoff (Q_s) over the maximum amount of rainfall that could be surface runoff ($W - V_I$) (Mishra & Singh, 2003).

$$\frac{V_R}{V_{max}} = \frac{Q_s}{W - V_I} \quad \text{Equation 9}$$

A water balance equation is also incorporated where it defines V_R according to Equation 10.

$$V_R = W - V_I - Q_s \quad \text{Equation 10}$$

An expression for Q_s is found by plugging Equation 10 and 8 into Equation 9 to get Equation 11.

$$Q_s = \frac{(W + \Delta S_i - \Delta S_{min})^2}{W + \Delta S_i - \Delta S_{min} + V_{max}} \quad \text{Equation 11}$$

Here precipitation (W) and initial storage (ΔS_i) vary, so in an effort to simplify the equation, effective precipitation is defined as the total precipitation plus initial storage (Equation 12).

$$W_{eff} = W + \Delta S_i \quad \text{Equation 12}$$

As a note, effective precipitation is typically defined as the precipitation after initial abstraction ($W - V_I$), but this is not how it is defined here. To find the precipitation after initial abstraction or W_I , ΔS_{min} must be subtracted from W_{eff} as seen in Equation 13. Since ΔS_{min} is constant for the watershed, the magnitude of the precipitation after initial abstraction will decrease the same amount relative to W_{eff} .

$$W_I = W - V_I = W - \Delta S_{min} + \Delta S_i = W_{eff} - \Delta S_{min} \quad \text{Equation 13}$$

Equation 12 can be plugged into the storm flow Equation 11 to produce the final equation (Equation 14), allowing for a single variable (W_{eff}) and two unknown constants (ΔS_{min} , V_{max}).

$$Q_s = \frac{(W_{eff} - \Delta S_{min})^2}{W_{eff} - \Delta S_{min} + V_{max}} \quad \text{iff } W_{eff} > \Delta S_{min} \quad \text{Equation 14}$$

V_{max} and ΔS_{min} can be combined into V_{range} defined as $V_{max} - \Delta S_{min}$. V_{range} represents the amount of storage available between the minimum storage level at which runoff is generated and to the point where the watershed cannot accept anymore into storage and all of the rainfall becomes runoff. V_{max} , ΔS_{min} , and V_{range} are parameters that characterize the relationship between catchment storage and event runoff, whose spatial patterns we would like to understand.

Now that storm flow is defined from a rain event and its initial storage, there needs to be a way to define this ΔS_i , and to define the separation between Q_s and Q_b . This is where the storage-discharge model will be used.

4 Storage-Discharge Relationship Theory

The method described below is based on that of Kirchner (2009). A storage-discharge relationship for a catchment can be used to infer relative storage in a watershed based on the discharge data. This allows the formulation of a first-order nonlinear differential equation that can predict streamflow hydrographs based off of precipitation and evapotranspiration inputs. The main assumption is that discharge depends only on the water stored in the basin, or that discharge is a function of storage as expressed in Equation 15.

$$Q = f(S) \quad \text{Equation 15}$$

The inverse is true as well, assuming that discharge is an increasing function of storage as expressed in Equation 16.

$$S = f^{-1}(Q) \quad \text{Equation 16}$$

Kirchner begins the derivation with the conservation of mass equation as listed in Equation 17 where S is storage, P is precipitation, E is evaporation, and Q is discharge with all variables averaged over the watershed.

$$\frac{dS}{dt} = P - E - Q \quad \text{Equation 17}$$

If Equation 15 is differentiated and combined with Equation 17 it yields Equation 18. The derivative of storage and discharge (dQ/dS) characterizes the fluctuations of the change in discharge to changes in storage.

$$\frac{dQ}{dt} = \frac{dQ}{dS} \frac{dS}{dt} = \frac{dQ}{dS} (P - E - Q) \quad \text{Equation 18}$$

Since storage cannot be directly measured, but is defined here as a function of discharge, a function can be developed through Equation 19 termed by Kirchner as the “sensitivity function”.

$$\frac{dQ}{dS} = f'(S) = f'(f^{-1}(Q)) = g(Q) \quad \text{Equation 19}$$

The $g(Q)$ function is estimated from observed flow data by combining Equations 18 and 19 to become Equation 20.

$$g(Q) = \frac{dQ}{dS} \cdot \frac{dt}{dt} = \frac{dQ/dt}{dS/dt} = \frac{dQ/dt}{P-E-Q} \quad \text{Equation 20}$$

This equation can be best estimated when precipitation and evapotranspiration are much smaller than discharge yielding Equation 21. This means that the sensitivity function can be determined solely from discharge data.

$$g(Q) = \frac{dQ}{dS} \approx \frac{-dQ/dt}{Q} \bigg|_{P \ll Q, E \ll Q} \quad \text{Equation 21}$$

Using discharge data, the values of $-dQ/dt$ and Q can be plotted on a log-log graph and fitted to Equation 22. $\ln(\bar{Q})$ is the log average of the Q s. Equation 22 has two parameters, b and a , where a has dimensions of time, and b is itself a function of discharge.

$$\ln \left(-dQ/dt \bigg|_{P \ll Q, E \ll Q} \right) \approx b(Q) (\ln(Q) - \ln(\bar{Q})) - \ln(a) \quad \text{Equation 22}$$

$$b(Q) = b_L + (b_U - b_L) \frac{1}{2} \left[1 + \operatorname{erf} \left(\frac{\ln(Q) - \ln(\bar{Q})}{\ln(\sigma_Q) \sqrt{2}} \right) \right] \quad \text{Equation 23}$$

Parameter b , shown in Equation 23, takes on the definition of a cumulative normal distribution function with an added intercept to shift the middle of the $\ln(Q)$ values and a scaling factor. b_U is defined as the upper slope fitting parameter and b_L as the lower slope fitting parameter. The upper and lower limits of b can be used to describe the storage-discharge relationship with an average exponent in Equation 24 and a curvature exponent in Equation 25.

$$\bar{b} = \frac{b_U + b_L}{2} \quad \text{Equation 24}$$

$$\Delta b = \frac{b_U - b_L}{2} \quad \text{Equation 25}$$

If the curvature is negative the shape of the sensitivity function is concave and if the curvature is positive the shape is convex. Both of these exponent values will be used to notice spatial trends of the storage-discharge relationship. The b from Equation 22 is constrained such that the slope can never be negative. This will help later in the analysis to keep storage values from going to extreme negatives. At first the data was fit to a quadratic, with some of the curves upper slopes turning negative, giving way to large, negative storage value. Equation 22 is used to force the slope positive. Once Equation 22 is minimized then the sensitivity function is found with Equation 26.

$$g(Q) = \left. \frac{-dQ/dt}{Q} \right|_{P \ll Q, E \ll Q} \approx \frac{\exp(\ln(-dQ/dt))}{Q} \quad \text{Equation 26}$$

The relative storage or the storage-discharge relationship can then be found by inverting Equation 19 as shown in Equation 27. Equation 27 will be used to estimate the value of ΔS_i which is used to define W_{eff} that is used in Equation 14.

$$\int dS = \int \frac{dQ}{g(Q)} \quad \text{Equation 27}$$

In many watersheds it is difficult to find periods where E is much smaller than Q . The approach can be modified to account for evaporation so long as E can be determined. The mass balance equation now only has precipitation going to zero as expressed in Equation 28.

$$\frac{dQ}{dt} = g(Q)(-E - Q) \quad \text{Equation 28}$$

Rearranging it becomes Equation 29.

$$-\frac{dQ}{dt} \bigg/ \left(1 + \frac{E}{Q}\right) = g(Q)Q \quad \text{Equation 29}$$

Another concept that will help in developing this theory is estimating a characteristic recession time constant (τ) defined in Equation 30 that describes how the discharge declines exponentially as a function of time (Equation 31) during recession states.

$$\tau = \frac{1}{g(Q)} \quad \text{Equation 30}$$

$$Q_b = Q_0 e^{-t/\tau} \quad \text{Equation 31}$$

Now the storage-discharge relationship can be practically tackled to develop a sensitivity function for the watersheds in a large database.

5 Details of Analysis

5.1 Data Description

This analysis uses a hydrologic data set compiled through the National Center for Atmospheric Research that compiles 671 basins across the continental United States (Newman et al., 2014).

Each basin has complete daily flow data from 1990 to 2009 and is a GAGES-II reference gage with less than 5% imperviousness and little human impact (Newman et al., 2014). The specific data used for this analysis is each basin's observed daily flow, precipitation, potential evapotranspiration (PET), and evapotranspiration (ET) all measured in mm/day as a volumetric depth (Newman et al., 2014). The PET is calculated with the Priestly-Taylor method (Newman et al., 2014).

5.2 Sensitivity function Regression Analysis

There is a flow chart of how the analysis is laid out Appendix A for reference that will be a helpful resource to read along with the following explanation. The analysis begins with developing the storage-discharge relationship based off of only the discharge data on days when there is no precipitation and evapotranspiration is 10 times less than the discharge in accordance with Equation 21. Estimating the $g(Q)$ equation from Equation 21 is found by plotting $-dQ/dt$ and Q on a log-log plot and fitting Equation 22 to a set of binned values (Kirchner, 2009). $-dQ/dt$ is found using Equation 32 and Q is found using Equation 33 with the truncated data set (Kirchner, 2009).

$$\frac{-dQ}{dt} = \frac{(Q_{t-\Delta t} - Q_t)}{\Delta t} \quad \text{Equation 32}$$

$$Q = \frac{(Q_{t-\Delta t} + Q_t)}{2} \quad \text{Equation 33}$$

The data points are binned because of the larger scatter occurring at the low flows in an effort to better represent the relationship with $-dQ/dt$ and Q (Kirchner, 2009). First, there are 100

evenly distributed bins according to Q . This means there are 100 sets of ranges between the minimum and maximum Q with data points $(-dQ/dt_i, Q_i)$ within the particular range in each set (Kirchner, 2009). Figure 4 shows a visual representation of what the bins look like evenly distributed.

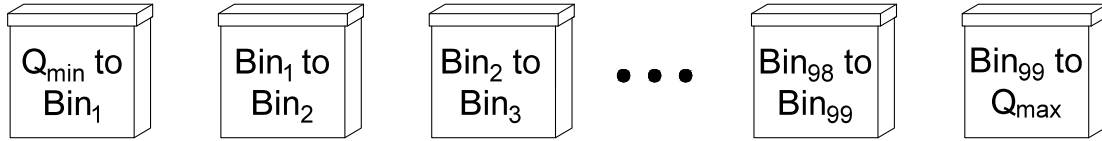


Figure 4 Bins evenly distributed

Of the data points in each bin, the mean and standard error of $-dQ/dt$ is calculated (Kirchner, 2009). Bins are then combined to meet set criteria. There must be at least three data points in a bin, and the standard error of $-dQ/dt$ must be less than half its mean as seen in Equation 34 (Kirchner, 2009).

$$\left[\left(\frac{-dQ}{dt} \right)_{std\ error} \right] < \frac{1}{2} \left[\left(\frac{-dQ}{dt} \right)_{mean} \right] \quad \text{Equation 34}$$

So, now the bins could look something like Figure 5 where you have, for example, three bins now combined into one bin. Note that the Q_{min} and Q_{max} are retained as the bounds. Basins will only be accepted for analysis if there are ten or more bins.

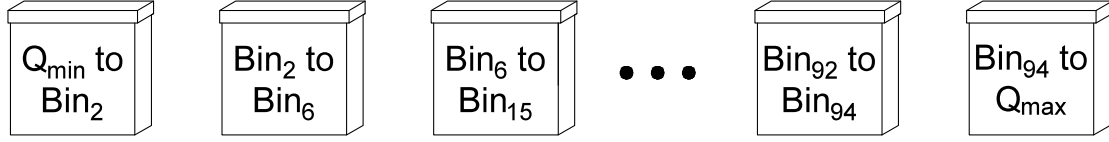


Figure 5 Bins with satisfactory standard error

When this is run across the 671 watersheds, about twenty percent of the watersheds have data points that meet the no precipitation and low PET rules while being able to meet the binning standard of producing ten or more bins. To improve the number of basins that a sensitivity function can be fit to, the remaining eighty percent of the basins are analyzed to include evaporation through Equation 29.

The bins can be subdivided in the same way described above except now the left hand side of Equation 29 is treated as $-dQ/dt$. If there are ten or more bins, then the left hand side of the equation and Q can be plotted with the new bins and the analysis can continue. So, anywhere in the following equations where there is a $-dQ/dt$ it is equivalent to the left hand side of Equation 29 when evaporation is included in the analysis. (Appendix A has a useful flow chart to visualize when evaporation is included and when it is not.) The function defined in Equation 22 is fitted to the log-log relationship of the binned Q 's and $-dQ/dt$'s, shown as the red dots in Figure 6.

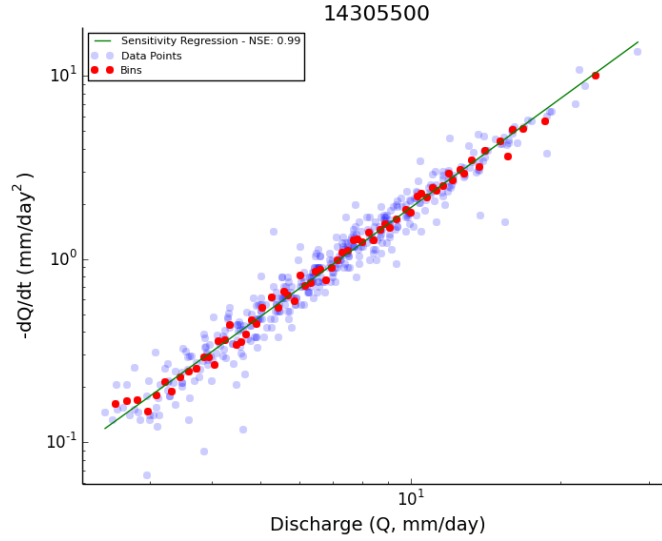


Figure 6 Storage-Discharge Relationship

Equation 22 is fit to the binned values through minimizing a weighted root mean square error ($RMSE_{wt}$) expressed in Equation 35.

$$RMSE_{wt} = \left[\frac{\sum w_i \left(\ln(-dQ_i/dt_i) - \ln(\overline{-dQ/dt}) \right)^2}{n} \right]^{1/2} \quad \text{Equation 35}$$

The weighting (w_i) is the inverse standard error for each bin, $(\overline{-dQ/dt})$ is the average and (n) is the number of bins. The weight lowers the impact of uncertain data points on the regression (Kirchner, 2009). Figure 6 shows an example of the scatter of the data points in blue, the bins in red, and Equation 22 as the blue line. Once Equation 22 is minimized then the sensitivity function is found with Equation 26.

5.3 Defining Storage

Values of $1/g(Q)$ are found along the range of Q by integration using the composite trapezoidal rule to find cumulative storage values for every Q . Then, according to each observed discharge the storage is interpolated to get a storage associated to each observation. Next, relative storage is defined as the storage value subtracted by the mean storage of the watershed. This defines negative storage as less than the mean, and positive storage as greater than the mean. Now that each day has a storage associated with it, each event has an initial and final storage associated with it as well.

5.4 Storm Flow Regression Analysis

Storms can now be analyzed and the parameters in Equation 14 can be estimated by minimizing the root-mean-square-error. Past storm events are selected for analysis if they have three days before and after of no rain, and that are large enough to cause an increase in discharge. The analysis only uses rain events that are large enough to cause an increase in discharge for two reasons. The first reason excludes small storms that do not affect discharge. The second reason accounts for snow events. It is assumed that snow will not have as great effect on discharge as rain because the snow will accumulate on the landscape and slowly contribute to runoff as it melts. Because of this, for many cases the stream's discharge will not be affected for snow events. So, the discharge data could be decreasing through a snow event. This analysis excludes any event where there is no increase in discharge after a rain/snow event.

Initial storage (ΔS_i) is defined as the storage on the day before it rained. To calculate final storage, the baseflow is found using the minimum discharge from the three days before it rains. It is the minimum of three days instead of the day before it rains because in many cases the

precipitation gage does not capture the full rain event. In many cases on the day before an event, the flow increases with no precipitation. This suggests it may have rained somewhere on the watershed where the gage is not capturing. So, the minimum of three days before the storm will better capture a true baseflow. So, the baseflow for an event is treated as a step-wise function with Q_0 as the minimum of the three days before the storm. The Q_0 defines the initial baseflow for the storm, and then it decreases exponentially with time throughout the event. The step-wise function starts at the time of Q_0 and steps forward in time to find the Q_b for each day of the event. Storm flow (Q_s) is the difference between the observed discharge and the baseflow. The final storage is then defined for this model as shown in Equation 36 as the storage when the storm discharge for that day is less than 10% of the cumulative storm discharge.

$$\Delta S_f = \Delta S_n \text{ where } Q_{s,n} < 0.10 \cdot \sum_{i=0}^n Q_{s,i} \quad \text{Equation 36}$$

Now that initial and final storage are known, initial abstraction can be found using Equation 8. Equation 14 can now be defined with the data. The precipitation data is known, initial storage has been defined, and all that is left is to define ΔS_{min} and V_{max} .

The initial guesses of the unknowns are set in the analysis to the following:

$$V_{max} = 0 \quad \text{Equation 37}$$

$$\Delta S_{min} = \Delta S_i + \frac{\min(\Delta S_i) - \text{mean}(\Delta S_i)}{2} \quad \text{Equation 38}$$

Now the function can be minimized according to the root mean square error (RMSE) defined as

$$RMSE = \sqrt{mean(Q_{s,obs})^2 - mean(Q_{s,pred})^2}$$

Equation 39

where $Q_{s,obs}$ is the storm flow found from the baseflow separation and $Q_{s,pred}$ is the predicted storm flow from Equation 14 with the fitted V_{max} and ΔS_{min} .

Once the function is minimized the model is complete and there is a working equation describing how the storage, and overland flow react to a given storm event for each watershed. Figure 7 is an example of the selected rain events for the watershed with their paired storm flow in the blue dots, and the fitted relationship in red. Now there are five parameters for each model to map: average exponent, exponent of curvature, ΔS_{min} , V_{max} , and V_{range} .

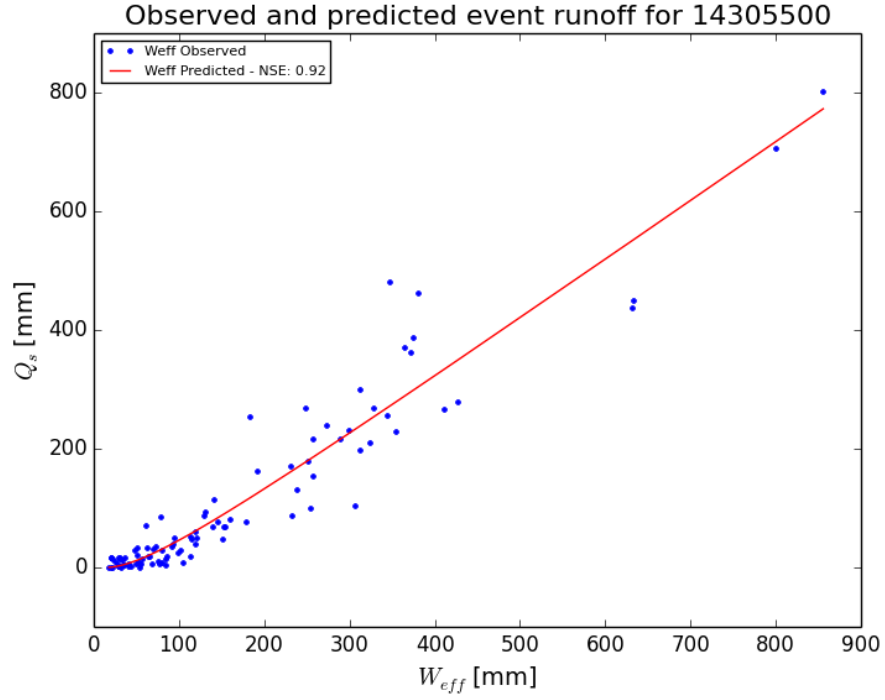


Figure 7 An example of observed and predicted runoff for a watershed

6 Results

Each of the 671 basins from the NCAR data set went through the analysis ten times each with a different optimal Sacramento Soil Moisture Accounting Model (SAC-SMA) parameter set that determined the PET and ET values. Each of the SAC-SMA parameter sets is determined using a shuffled complex evolution algorithm (SCE) global optimization routine (Newman et al., 2014). The sensitivity of the optimal sets to V_{max} and ΔS_{min} was mostly small for all but one of the sets. So, to reconcile which parameter set to choose, the set with median value of V_{range} was used. Any analysis henceforth only uses the specific optimal parameter set for its watershed.

A better efficiency to represent the model compared to that of the Nash-Sutcliffe Efficiency (NSE) uses a long-term monthly mean flow instead of a mean flow over the total 30 years of data (Newman et al., 2014). This is calculated by finding the mean flow ($Q_{m,i}$) for every month, where a month is defined as a 31-day window. Q_m is represented as averaging each mean flow over all of the mean flows for that specific 31-day window across the 30 years as seen in Equation 40.

$$Q_m = \frac{1}{n} (Q_{m,1} + Q_{m,2} + \dots + Q_{m,n})$$

Equation 40

where n is the number of years

The modified NSE can now be found using Equation 43 where F_1^2 and F_m^2 are represented as Equation 41 and 42 respectively (Garrick, Cunnane, & Nash, 1978).

$$F_1^2 = \sum (Q_{obs} - Q_{model})^2$$

Equation 41

$$F_m^2 = \sum (Q_{obs} - Q_m)^2$$

Equation 42

$$NSE = \frac{F_m^2 - F_1^2}{F_m^2}$$

Equation 43

Figure 8 and Figure 9 show the break down of how each watershed performed based off of the NSE.

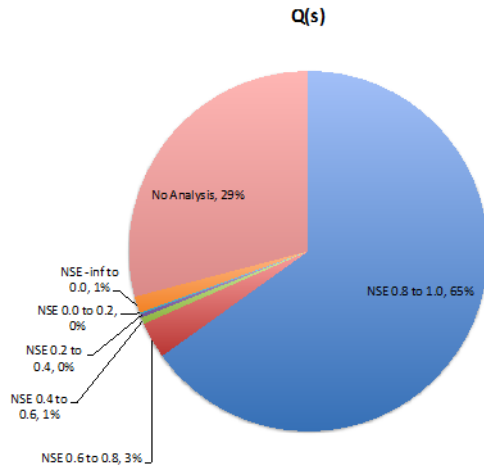


Figure 8 NSE values of each basin for the sensitivity function

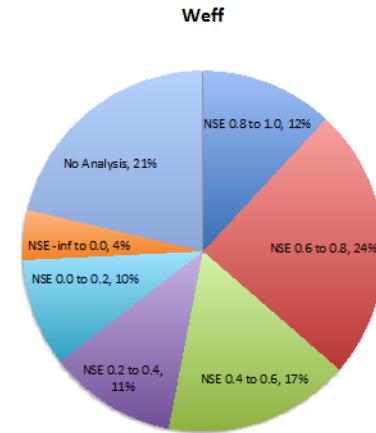


Figure 9 NSE values of each basin for the storm flow function

Figure 10 and Figure 12 are example graphs of the storage-discharge relationship for the analysis with no precipitation and minimal evapotranspiration and the analysis including evapotranspiration respectively. The blue dots represent each data point, the red dots represent the binned values, and the green line is the sensitivity function that is fit to the binned red dots. Figure 11 and Figure 13 are their respective storm flow equation. The blue dots represent storm events and the red line is the equation. Each of these is an example of what was performed for each basin in the data set.

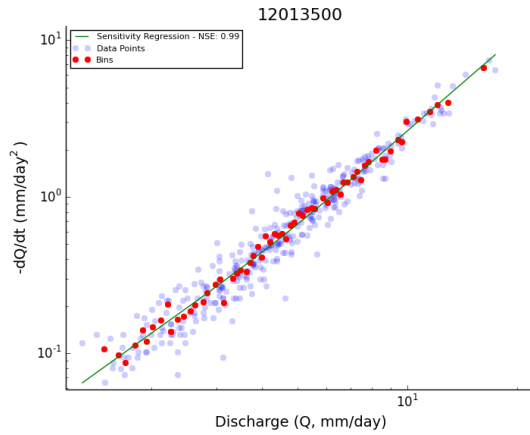


Figure 10 Storage-Discharge relationship with no precipitation and minimal evapotranspiration for Basin 12013500

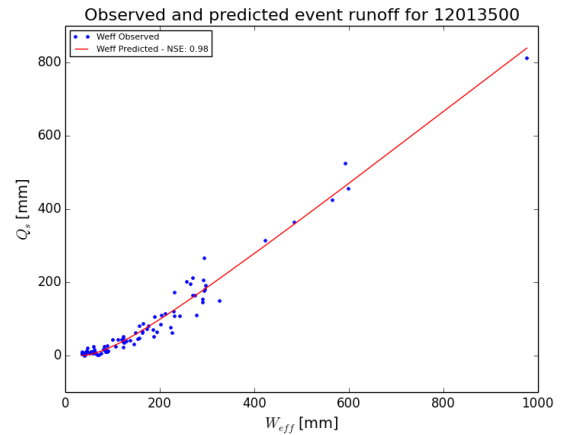


Figure 11 predicted storm flow for Basin 12013500

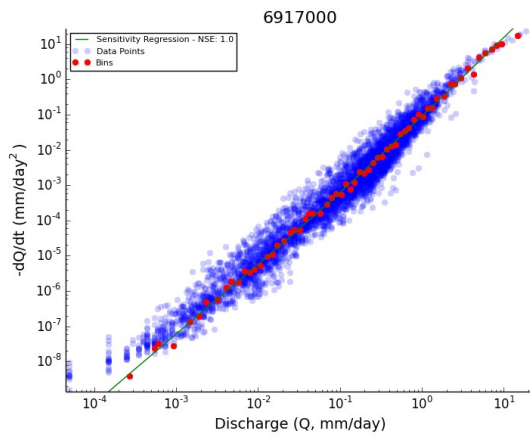


Figure 12 Storage-Discharge relationship with no precipitation and including evapotranspiration for Basin 6917000

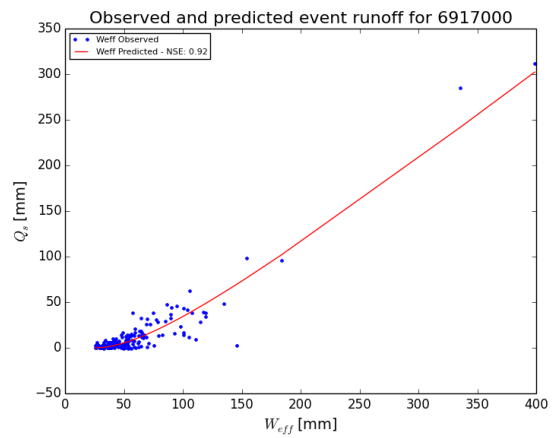


Figure 13 predicted storm flow for Basin 6917000

Each of the 477 basins that are able to achieve a storage discharge relationship is plotted in various ways to explore potential spatial patterns. The NSE values for the storage discharge relationship for the sensitivity function and the storm flow prediction are plotted spatially in Figure 14 and Figure 15 respectively. Keeping these two maps in mind to look at the next few maps is key in the confidence placed in the patterns seen.

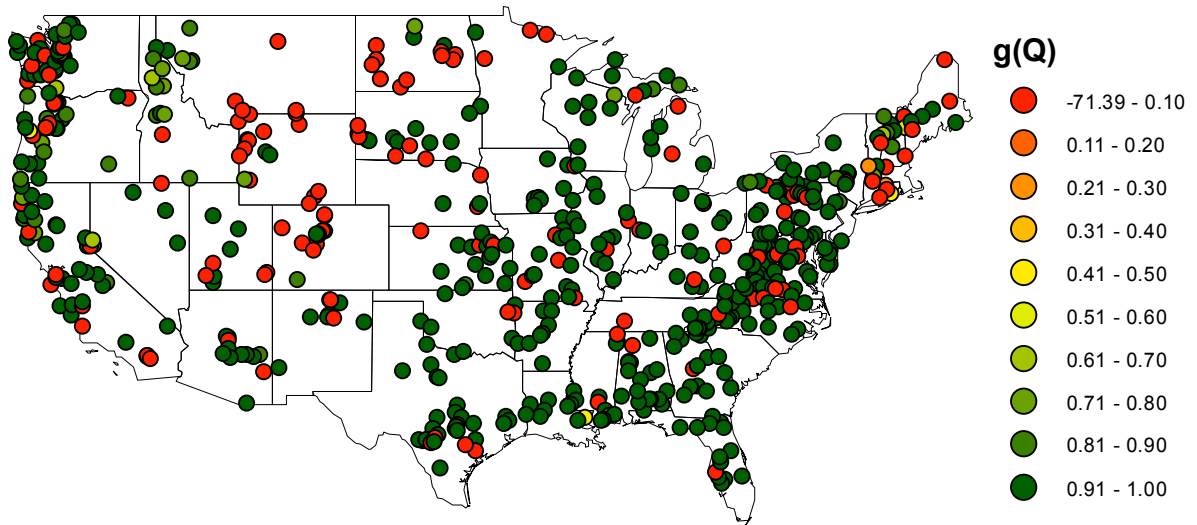


Figure 14 NSE values mapped spatially for the sensitivity function for every basin that met the analysis criteria

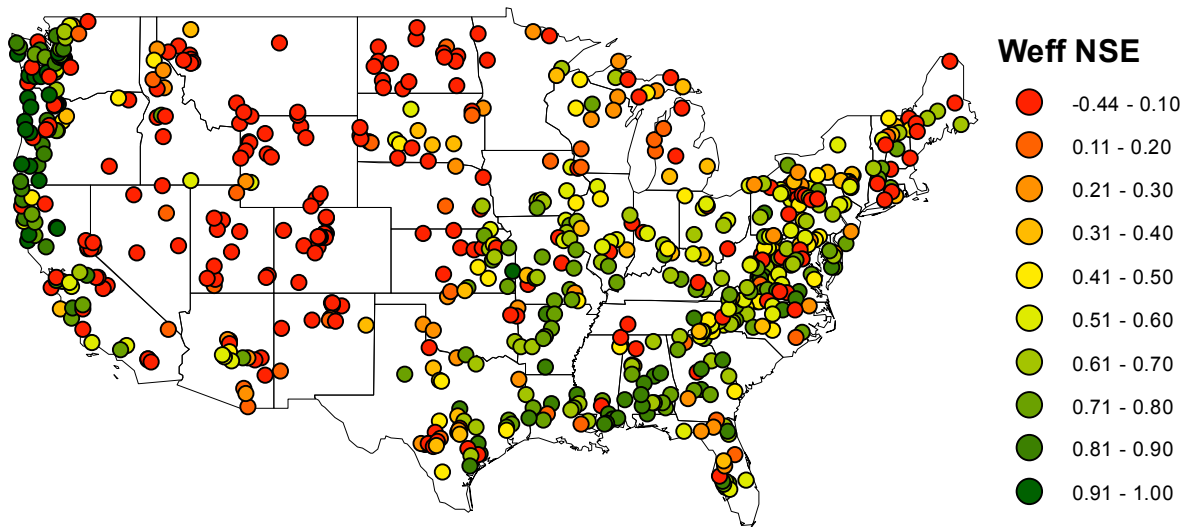


Figure 15 NSE values mapped spatially for the storm flow final analysis

Figure 16 and Figure 17 show the exponent average and exponent curvature for the sensitivity function. If the exponent curvature is positive then the sensitivity function is concave up. It becomes linear as it reaches zero, and moves to concave down when negative.

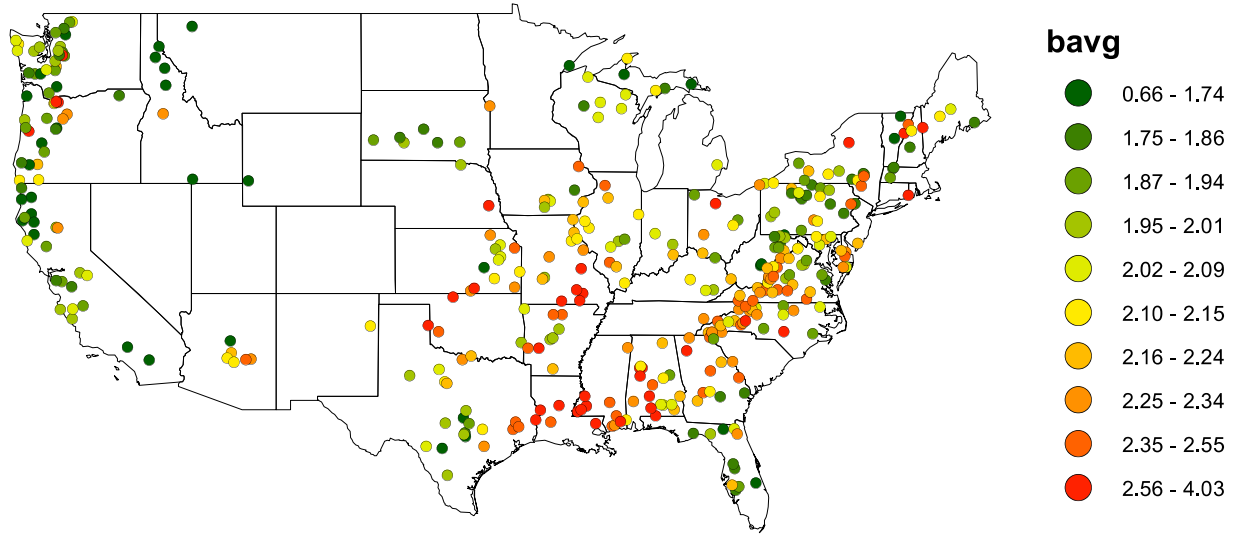


Figure 16 Exponent average parameter for the sensitivity function

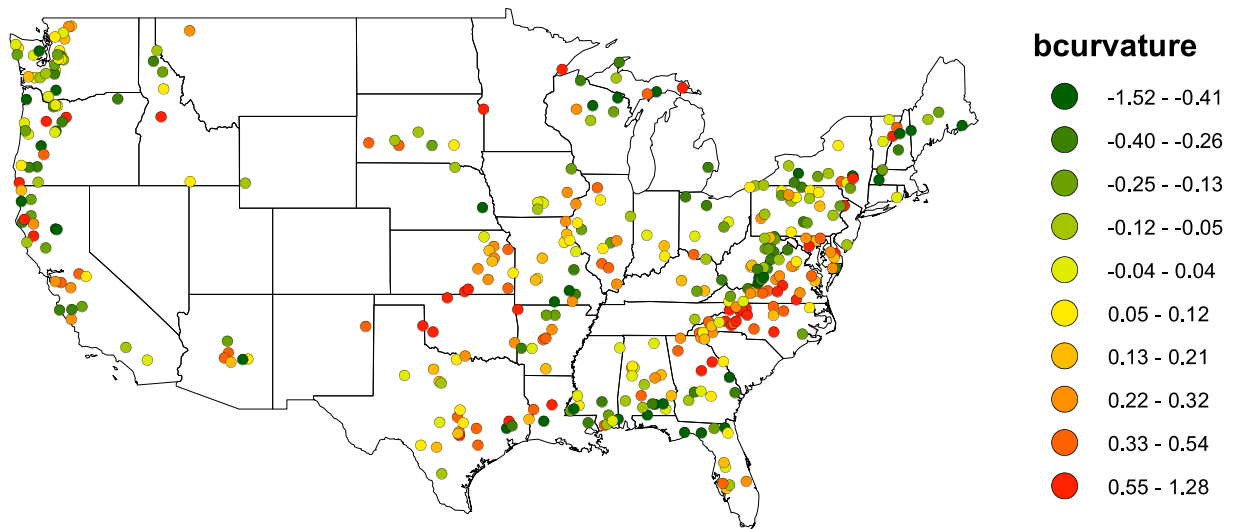


Figure 17 Exponent curvature for the sensitivity function

Figure 18, Figure 19, and Figure 20 shows the spatial trends for V_{max} , ΔS_{min} , and V_{range} respectively. Because of the large range of values, the values are distributed with an equal number of units in each category.

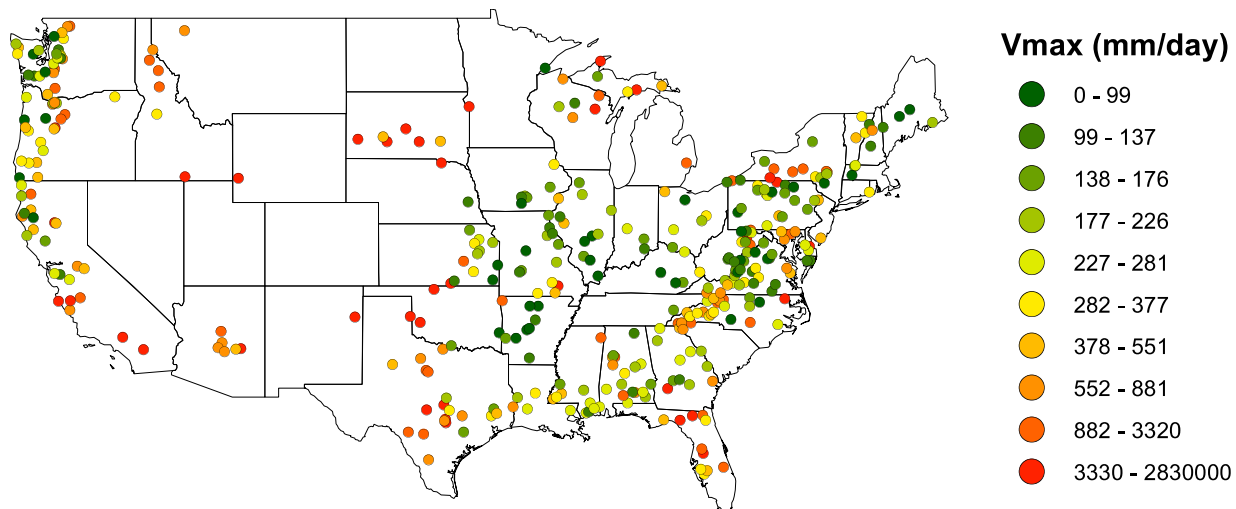


Figure 18 Maximum storage value for each basin mapped spatially

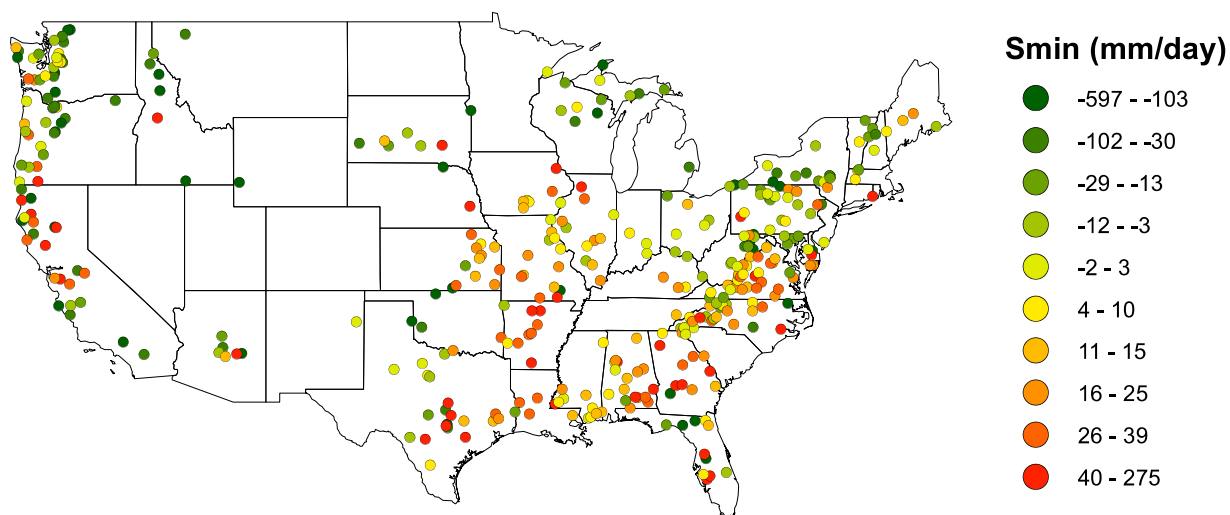


Figure 19 Minimum storage value for each basin mapped spatially

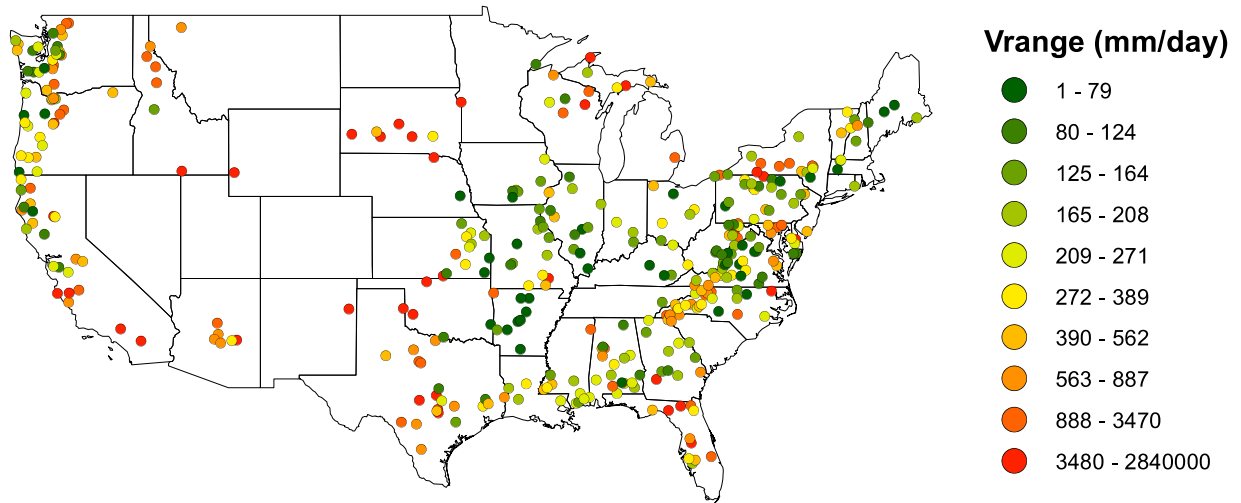


Figure 20 Total storage for each basin mapped spatially

Figure 21 and Figure 22 show a blown up image of the Appalachian Mountains so that more noticeable trends can be observed. The sizes of the circles are related to the NSE values, so the larger the circle, the larger the NSE value. This will help to visually see the confidence to put in the described patterns.

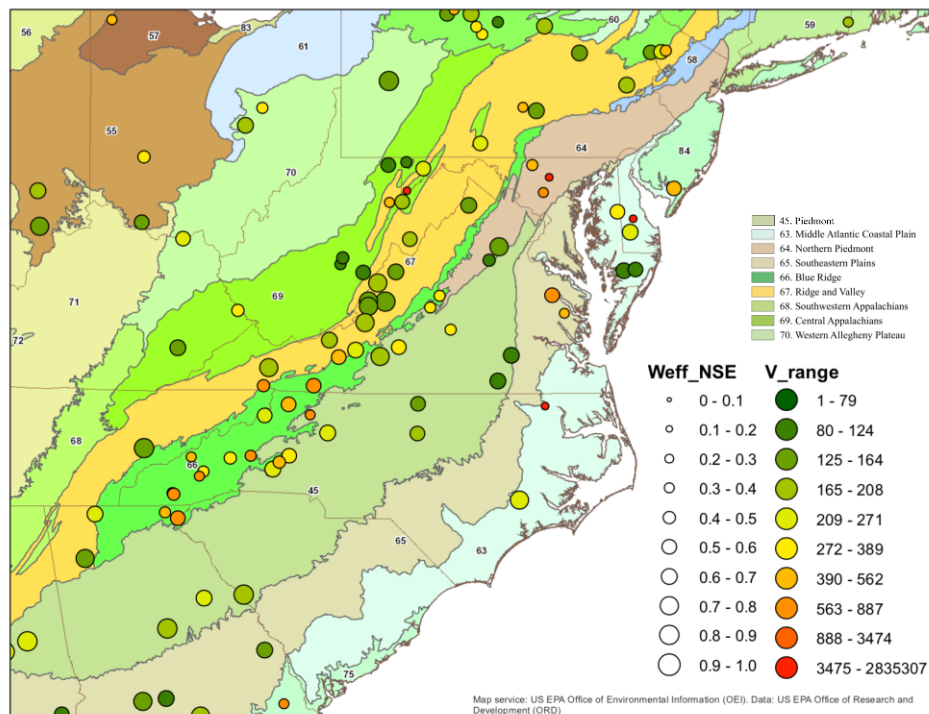


Figure 21 Vrange values along the Appalachian Mountains

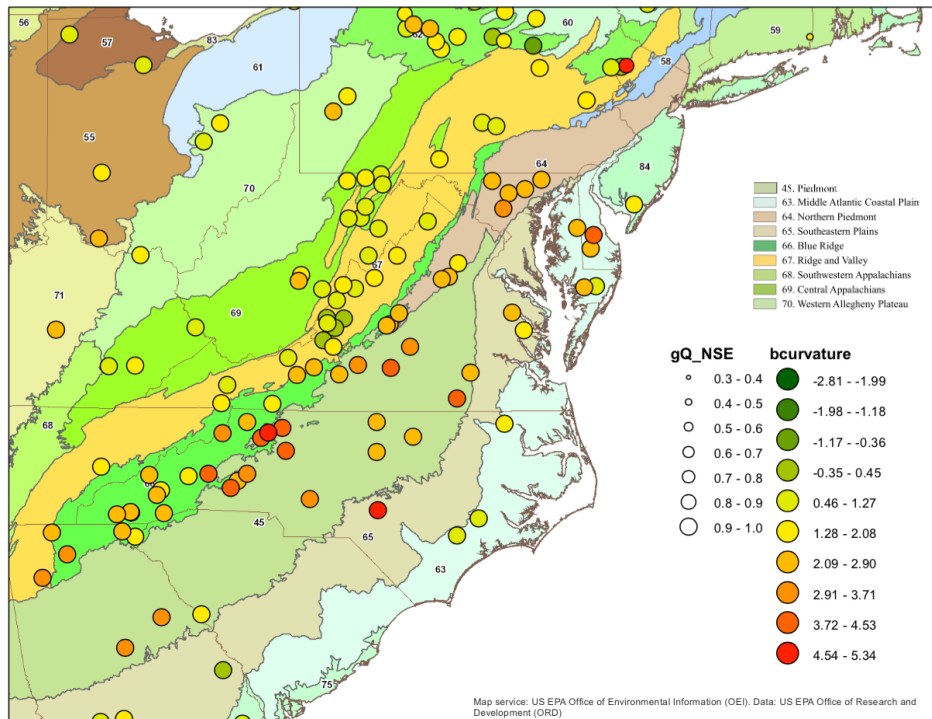


Figure 22 bcurvature values along the Appalachian Mountains

The aridity index is defined as the ratio of the annual PET to the annual precipitation average over all years. The aridity index was plotted with the various parameters, and of them all only V_{range} shows a small hint to a relationship to the aridity index as shown in Figure 23.

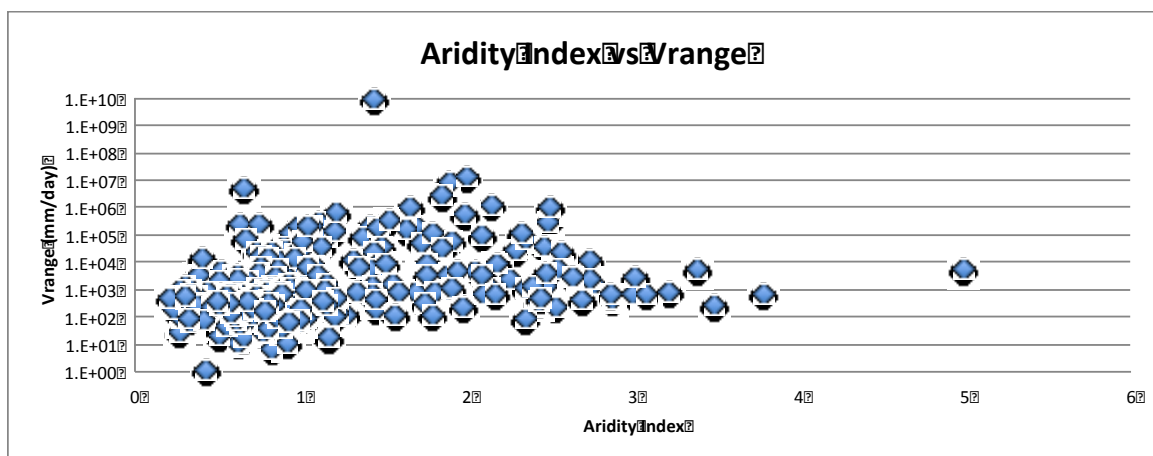


Figure 23 Plot of aridity index and V_{range} to see any correlation

7 Discussion

The sensitivity functions' NSE values shown in Figure 14 do a great job across almost all of the watersheds with only a few of them performing below a 0.50 NSE as shown in green. The northwestern and far northeast areas of the US give poor NSE values for the sensitivity function, along with areas around the Great Lakes. Interestingly, the West Coast does the best overall for NSE values concerning the final storm flow analysis shown in Figure 15. The central and western part of the US excluding the West Coast does a poor job with predicting storm flow. The southeast and eastern US do a reasonable job with above a 0.50 NSE overall.

Figure 16 shows consistent values across the West Coast, as well as in the Gulf of Mexico around Louisiana and Mississippi for the exponent average parameter. The values across the Appalachian Mountains are also fairly consistent in the upper half range. The exponent of curvature shown in Figure 17 is not as consistent across the West Coast as the exponent average parameter is. The coastline around the Gulf of Mexico stays within negative curvature values. The values along the Appalachian Mountains seem to shift from positive to negative values abruptly around Virginia, showing a different pattern from the exponent average parameter. Figure 18 shows that maximum storage values. The largest values appear to be in the central western part of the US, but these watersheds are also associated with poor NSE values so should be neglected. There appears to be regional patterns suggesting that this value could be a good classification parameter. For example, low maximum storage is clustered around Iowa, Illinois, Missouri, and Arkansas. Washington has a cluster of low storage values around the coastal area. The pattern around the Appalachian Mountains is also distinct here as well with

high maximum storage values in the southern states of the mountain range with a sharp contrast to low maximum storage values in the northern states of the mountain range. Figure 19 shows the minimum storage values across all the basins. This parameter shows clusters, but not necessarily the same ones. For example, the pattern along the Appalachian Mountains is no longer apparent. However patterns in the south in states like Louisiana, Mississippi, and Alabama they all have high minimum storage values. Figure 20 combines maximum and minimum storage into the range parameter. This shows similar trends with the maximum storage values because the magnitude is larger compared to the minimum storage. So, the pattern along the Appalachian Mountains is still apparent, along with the cluster of low total storage around Iowa, Illinois, Missouri, and Arkansas. The southern states also tend to cluster around smaller total storage values.

Figure 21 and Figure 22 explore in more detail the patterns of the curvature parameter and the total storage along the Appalachian Mountains. First, look at Figure 22 and notice the shift as the curvature parameter goes from positive to negative going north along the range. The NSE values are consistently high here confirming the pattern. Figure 21 shows a similar trend along the range as well with the value decreasing north along the range. However, the NSE values are not as strong, particularly for the southern states of the range with NSE values around 0.3 to 0.6. Both parameters are overlain by ecoregions that represent commonalities between “geology, physiography, vegetation, climate, soils, land use, wildlife, and hydrology” (U.S. Environmental Protection Agency, 2013). It appears that the total storage and curvature values make the transition from the Blue Ridge Mountains (66) to the Ridge and Valley Appalachians (67). Hard bedrock, steep slopes, and high rainfall characterize the Blue Ridge Mountains,

while soft sedimentary rock covered by forests characterize the Ridge and Valley Appalachians (Jackson & Stakes 2004). This could mean that these parameters could be used alongside descriptions of the landscape such as soil or geology to aid in classification of how a watershed will respond within the region in regards to storage and storm flow.

The aridity index was plotted against the five parameters but revealed no correlation except for a small correlation to V_{range} . This means that the parameters are not affected too much by the climate. Figure 23 is plotted as a semi-log graph to better show the distribution of V_{range} because of the large range. There is a gradual increase in the aridity index with V_{range} suggesting that the drier the climate, the larger the V_{range} . This is not necessarily consistent because it could be inferred that evaporation would be much stronger in arid areas causing less storage. So, for the most part the parameters are unaffected by climate, but rather affected through descriptions such as the topography or soil.

Overall, the five parameters shown in the maps are a step in the right direction. Most of the basins successfully completed the analysis, and gave inference into how to classify watersheds after being plotted spatially.

8 Conclusion

In this analysis, Kirchner's storage discharge relationship and the proportionality rule from the SCS-CN method was used together to model store/release of water and storm flow across 671 basins. The parameters were plotted on a map to see any spatial trends. This simple model provided sufficient information to notice spatial trends, such as the discussed pattern along the

Appalachian Mountains. The past discharge data helped in locating these spatial patterns. Two distinguishing characteristics among catchments that hydrologists are interested in is storm flow and how catchments store or release water. This analysis is able to combine both based on historic data to help see patterns that will lead to catchment classification. Going forward, there should be deeper study to find out what attributes of a landscape, such as topography or soil compositions, contribute the most to the pattern changes with storm flow and the storing/releasing of water.

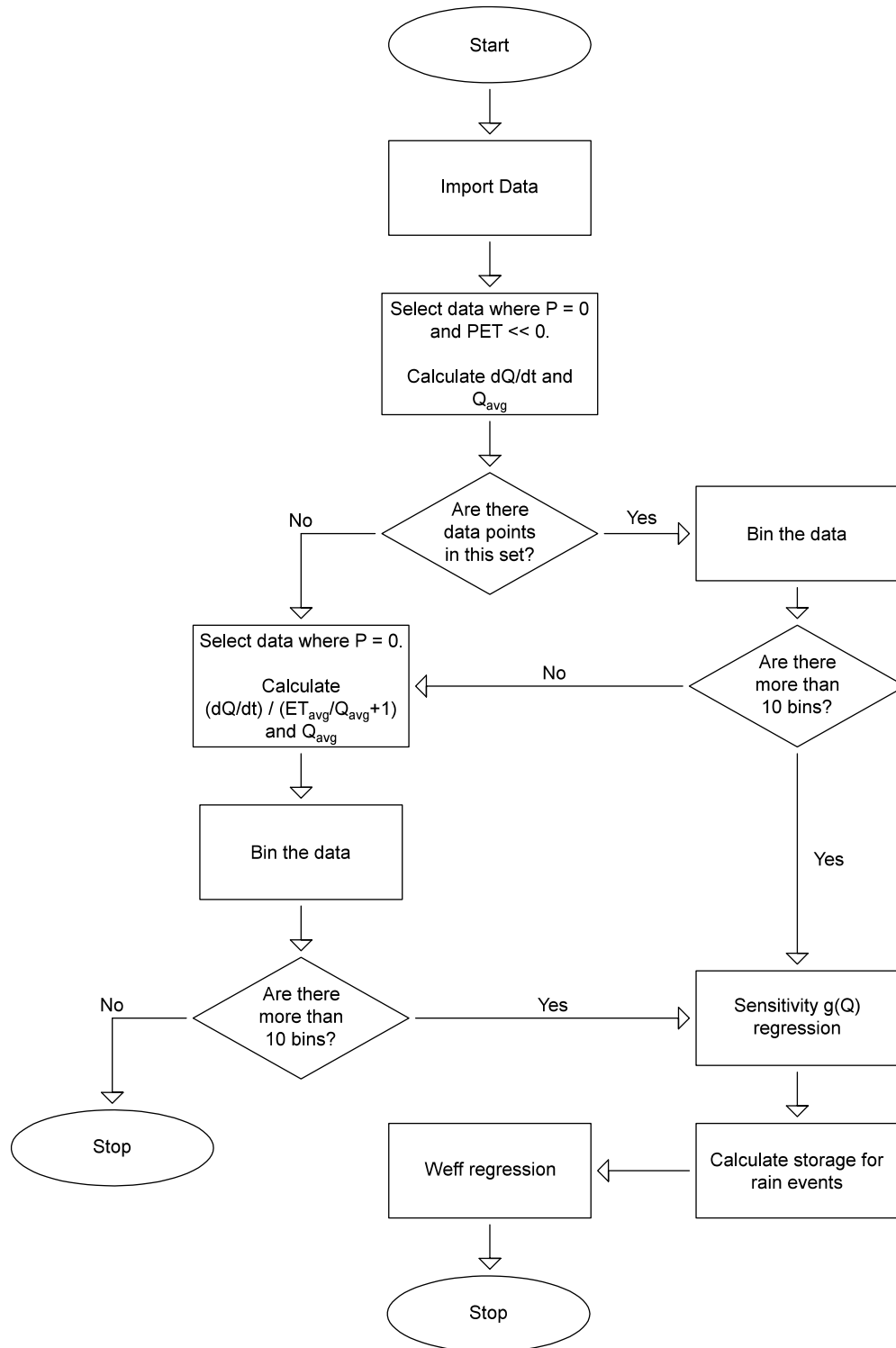
9 References

- Bosznay, M. (1989). Generalization of SCS Curve Number Method, 115(1), 139–144.
- Boughton, W. C. (1989). A Review of the USDA SCS Curve Number Method. *Australian Journal of Soil Research*, 27, 511–523.
- Brutsaert, W., & Nieber, J. L. (1977). Regionalized drought flow hydrographs from a mature glaciated plateau. *Water Resources Research*, 13(3), 637. <http://doi.org/10.1029/WR013i003p00637>
- Dingman, S. L. "Chapter 9: Stream Response to Water-Input Events." *Physical Hydrology*. Long Grove, IL: Waveland, 2002. 445-49. Print.
- Garrick, M., Cunnane, C., & Nash, J. E. (1978). A criterion of efficiency for rainfall-runoff models. *Journal of Hydrology*, 36(3-4), 375–381. [http://doi.org/10.1016/0022-1694\(78\)90155-5](http://doi.org/10.1016/0022-1694(78)90155-5)
- Harman, C., & Troch, P. a. (2014). What makes Darwinian hydrology “darwinian”? Asking a different kind of question about landscapes. *Hydrology and Earth System Sciences*, 18(2), 417–433. <http://doi.org/10.5194/hess-18-417-2014>
- Jackson, Ed and Stakes, Mary, *The Georgia Studies Book: Our State and Nation*, Carl Vinson Institute of Government, University of Georgia, 2004.
- Kirchner, J. W. (2009). Catchments as simple dynamical systems: Catchment characterization, rainfall-runoff modeling, and doing hydrology backward. *Water Resources Research*, 45(2), 1–34. <http://doi.org/10.1029/2008WR006912>

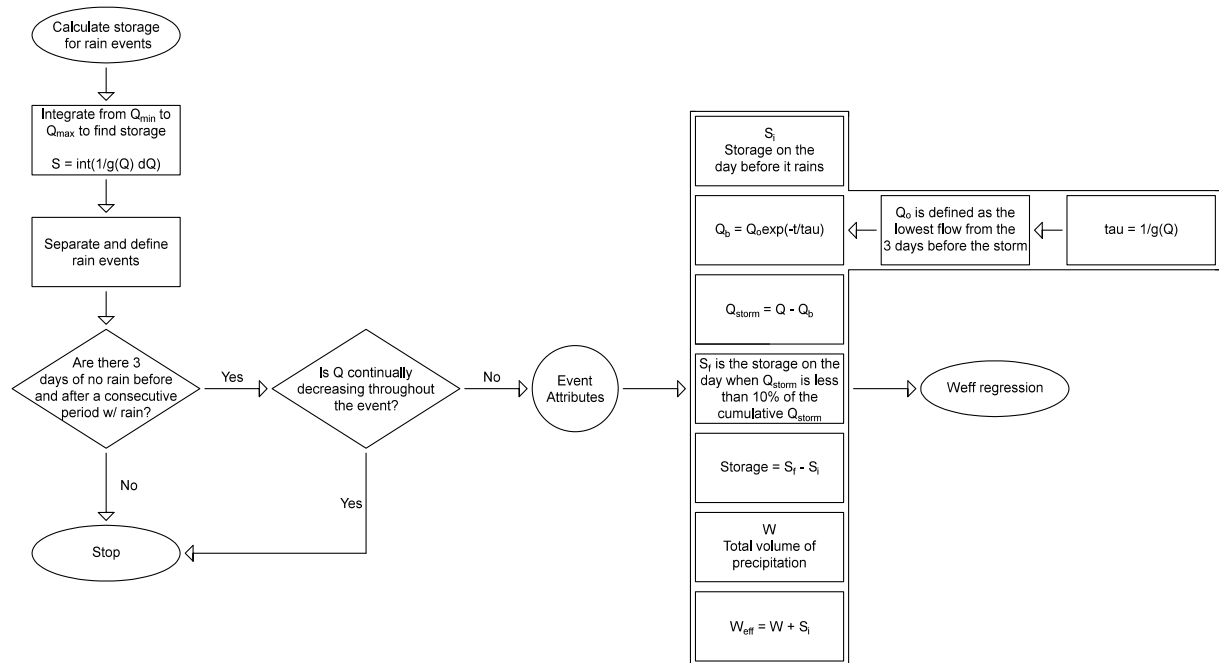
- Mishra, S. K., & Singh, V. P. (2003). Soil Conservation Service Curve Number (SCS-CN) Methodology, 513. <http://doi.org/10.1007/978-94-017-0147-1>
- Newman, a. J., Clark, M. P., Sampson, K., Wood, a., Hay, L. E., Bock, a., ... Duan, Q. (2014). Development of a large-sample watershed-scale hydrometeorological dataset for the contiguous USA: dataset characteristics and assessment of regional variability in hydrologic model performance. *Hydrology and Earth System Sciences Discussions*, 11(5), 5599–5631. <http://doi.org/10.5194/hessd-11-5599-2014>
- Plummer, a, & Woodward, D. (1998). The origin and derivation of Ia/S in the runoff curve number system. *International Water Resources Engineering Conference*, 1260–1265.
- Wagener, T., Sivapalan, M., Troch, P., & Woods, R. (2007). Catchment Classification and Hydrologic Similarity. *Geography Compass*, 1, 1–31. <http://doi.org/10.1111/j.1749-8198.2007.00039.x>
- Wittenberg, H., & Sivapalan, M. (1999). Watershed groundwater balance estimation using streamflow recession analysis and baseflow separation. *Journal of Hydrology*, 219(1-2), 20–33. [http://doi.org/10.1016/S0022-1694\(99\)00040-2](http://doi.org/10.1016/S0022-1694(99)00040-2)
- Woodward, D., Hawkins, R., Hjelmfelt, A., Mullem, J. A., & Quan, Q. (2002). Curve Number Method: Origins, Applications and Limitations. Derwood: U.S. Geological Survey Advisory Committee on Water Information - Second Federal Interagency Hydrologic Modeling Conference.

10 Appendix A: Flow Charts

Master Code Flow Chart



Rain Event Code (fe_find_event_storage) Flow Chart



11 Appendix B: Code

Code for - Master

This is the master code that will call subcodes to perform the analysis

```
import pandas as pd
import numpy as np
import os, shutil
import fa_getdata
import fb_dQ_dt_AND_avg_discharge
import fc_Binning
import fd_sensitivity_regression
import fe_find_event_storage
import fg_least_square_Qs_regression
import fh_dQ_dt_AND_evap_term
```

Define file directories

```
maindirectory = "/Documents/Research/Python"
graph_directory = "/Documents/Research/Python/Graph"
raindata_directory = "/Documents/Research/Python/Rain Data"
summary_directory = "/Documents/Research/Python/Summary of Rain Events"
Weff_directory = "/Documents/Research/Python/Weff Graph"
NSE_08to10 = "/Documents/Research/Python/Weff Graph/NSE 0.8 to 1.0"
NSE_06to08 = "/Documents/Research/Python/Weff Graph/NSE 0.6 to 0.8"
NSE_04to06 = "/Documents/Research/Python/Weff Graph/NSE 0.4 to 0.6"
NSE_02to04 = "/Documents/Research/Python/Weff Graph/NSE 0.2 to 0.4"
NSE_00to02 = "/Documents/Research/Python/Weff Graph/NSE 0.0 to 0.2"
NSE_negative = "/Documents/Research/Python/Weff Graph/NSE -inf to 0.0"
G_NSE_08to10 = "/Documents/Research/Python/Graph/NSE 0.8 to 1.0"
G_NSE_06to08 = "/Documents/Research/Python/Graph/NSE 0.6 to 0.8"
G_NSE_04to06 = "/Documents/Research/Python/Graph/NSE 0.4 to 0.6"
G_NSE_02to04 = "/Documents/Research/Python/Graph/NSE 0.2 to 0.4"
G_NSE_00to02 = "/Documents/Research/Python/Graph/NSE 0.0 to 0.2"
G_NSE_negative = "/Documents/Research/Python/Graph/NSE -inf to 0.0"
```

```
model = [5,11,27,33,59,66,72,80,94] # ID numbers for the SAC-SMA parameter sets
site_list = pd.read_csv('site_list_mean.csv', dtype={'SiteID': str}) # Reads a list containing names of the 671 basins and the SAC-SMA parameter set to be used for each basin's analysis
```

Collect information on each watershed into a data frame - initialize

```
watersheds = site_list.copy()
placement = np.zeros(len(site_list))
```

```

watersheds['No of data points'] = placement.copy()
watersheds['No of bins'] = placement.copy()
watersheds['No of rain events'] = placement.copy()
watersheds['V_max'] = placement.copy()
watersheds['S_min'] = placement.copy()
watersheds['V_max < S_min?'] = placement.copy()
watersheds['g(Q) NSE'] = placement.copy()
watersheds['RMSE'] = placement.copy()
watersheds['a'] = placement.copy()
watersheds['bL'] = placement.copy()
watersheds['bU'] = placement.copy()
watersheds['min flow'] = placement.copy()
watersheds['max flow'] = placement.copy()
watersheds['mean storage'] = placement.copy()
watersheds['Evaporation included?'] = placement.copy()
watersheds['Empty data set?'] = placement.copy()
watersheds['Weff NSE'] = placement.copy()

```

```

for q in range(len(model)):

```

```

    # List of all basins to be used with specific parameter set q
    model_list = site_list[site_list['Model']==model[q]]
    # Change directory to access basin data under parameter set q
    modeldirectory = "/Documents/Research/Python/Model %s" % model [q]
    os.chdir(modeldirectory) # Change the directory
    retval = os.getcwd() # Check current working directory
    print "Directory changed successfully %s" % retval

```

```

    # Initialize data frames

```

```

    RainData = np.zeros(len(model_list),dtype=object)
    Summary_of_rain_events = np.zeros(len(model_list),dtype=object)
    Weff_Graph_08to10 = np.zeros(len(site_list),dtype=object)
    Weff_Graph_06to08 = np.zeros(len(site_list),dtype=object)
    Weff_Graph_04to06 = np.zeros(len(site_list),dtype=object)
    Weff_Graph_02to04 = np.zeros(len(site_list),dtype=object)
    Weff_Graph_00to02 = np.zeros(len(site_list),dtype=object)
    Weff_Graph_negative = np.zeros(len(site_list),dtype=object)
    Graph_08to10 = np.zeros(len(site_list),dtype=object)
    Graph_06to08 = np.zeros(len(site_list),dtype=object)
    Graph_04to06 = np.zeros(len(site_list),dtype=object)
    Graph_02to04 = np.zeros(len(site_list),dtype=object)
    Graph_00to02 = np.zeros(len(site_list),dtype=object)
    Graph_negative = np.zeros(len(site_list),dtype=object)

```

```

for i in range(len(model_list)):

```

```

    this_site = model_list.ix[model_list.index[i]] # Calls a specific basin within the parameter set
    data = fa_getdata.get_timeseries_data(this_site['File Name']) # Data for the basin
    if len(data) <= 1:

```

```

print 'Empty data set'
watersheds['Empty data set?'][model_list.index[i]] = 'yes'
else:
    events = fa_getdata.get_timeseries_data(this_site['File Name'])
    # Calculates Q and dQ/dt
    avg_discharge, dQ_dt = fb_dQ_dt_AND_avg_discharge.dQ_dt_AND_avg_discharge(data, model_list, watersheds, i)

    if len(avg_discharge) <= 1:
        # Q and dQ/dt including evaporation
        avg_discharge, dQ_dt_EQone = fh_dQ_dt_AND_evap_term.dQ_dt_AND_evap_term(model_list, data, watersheds, i)
        # Calculates the bins under the specific criteria
        Bins, mean_bin, stderr_dQ_dt = fc_Binning.Binning(model_list, avg_discharge, dQ_dt_EQone, watersheds, i)

    if (len(Bins)-1) >= 10:
        # Perform sensitivity regression
        coeff, NSE, dQdt_eq = fd_sensitivity_regression.sensitivity_regresssion(model_list, events, Bins, mean_bin, stderr_dQ_dt, avg_discharge, dQ_dt_EQone, this_site,
watersheds,i)
        # Finds rain events that meet criteria
        rainevents = fe_find_event_storage.find_event_storage(dQdt_eq, model_list, coeff, events, data, this_site, watersheds, i)
        watersheds['No of rain events'][model_list.index[i]] = len(rainevents)

    if len(rainevents) > 10:
        # Perform regression to find Vmax and Smin for the basin
        V_max, S_min, RMSE, Weff_NSE = fg_least_square_Qs_regression.least_square_Qs_regression(model_list, rainevents, watersheds, i, this_site)

        print '----- Watershed ',this_site['Basin_ID'],'i =',i,'-----'
        print ''
        print ' V_max =',V_max.round(2), ' S_min =',S_min.round(2), ' RMSE =', RMSE.round(2)
        print ' Q_s = [(W_eff - (',S_min.round(2),'))^2] / [', V_max.round(2), ' + W_eff - (',S_min.round(2),')]'
        print ''
    else:
        print '----- Watershed ',this_site['Basin_ID'],'i =',i,'-----'
        print ''
        print 'Not enough rain events'
        print ''
    watersheds['Evaporation included?'][model_list.index[i]] = 'yes'

elif len(avg_discharge) != 0:
    # Calculates the bins under the specific criteria
    Bins, mean_bin, stderr_dQ_dt = fc_Binning.Binning(model_list, avg_discharge, dQ_dt, watersheds, i)

    if (len(Bins)-1) >= 10:
        # Perform sensitivity regression
        coeff, NSE, dQdt_eq = fd_sensitivity_regression.sensitivity_regresssion(model_list, events, Bins, mean_bin, stderr_dQ_dt, avg_discharge, dQ_dt, this_site, watersheds,i)
        # Finds rain events that meet criteria
        rainevents = fe_find_event_storage.find_event_storage(dQdt_eq, model_list, coeff, events, data, this_site, watersheds, i)
        watersheds['No of rain events'][model_list.index[i]] = len(rainevents)

```

```

if len(rainevents) > 10:
    # Perform regression to find Vmax and Smin for the basin
    V_max, S_min, RMSE, Weff_NSE = fg_least_square_Qs_regression.least_square_Qs_regression(model_list, rainevents, watersheds, i, this_site)

    print '----- Watershed ',this_site['Basin_ID'],'i =',i,'-----'
    print ''
    print ' V_max =',V_max.round(2), ' S_min =',S_min.round(2), ' RMSE =', RMSE.round(2)
    print ' Q_s = [(W_eff - (' ,S_min.round(2),'))^2] / [', V_max.round(2), ' + W_eff - (' ,S_min.round(2),')]'
    print ''
else:
    print '----- Watershed ',this_site['Basin_ID'],'i =',i,'-----'
    print ''
    print 'Not enough rain events'
    print ''
watersheds['Evaporation included?'][model_list.index[i]] = 'no'
elif (len(Bins)-1) <= 10:
    # Q and dQ/dt including evaporation
    avg_discharge, dQ_dt_EQone = fh_dQ_dt_AND_evap_term.dQ_dt_AND_evap_term(model_list, data, watersheds, i)
    # Calculates the bins under the specific criteria
    Bins, mean_bin, stderr_dQ_dt = fc_Binning.Binning(model_list, avg_discharge, dQ_dt_EQone, watersheds, i)

    if (len(Bins)-1) >= 10:
        # Perform sensitivity regression
        coeff, NSE, dQdt_eq = fd_sensitivity_regression.sensitivity_regresssion(model_list, events, Bins, mean_bin, stderr_dQ_dt, avg_discharge, dQ_dt_EQone, this_site,
watersheds,i)
        # Finds rain events that meet criteria
        rainevents = fe_find_event_storage.find_event_storage(dQdt_eq, model_list, coeff, events, data, this_site, watersheds, i)
        watersheds['No of rain events'][model_list.index[i]] = len(rainevents)

    if len(rainevents) > 10:
        # Perform regression to find Vmax and Smin for the basin
        V_max, S_min, RMSE, Weff_NSE = fg_least_square_Qs_regression.least_square_Qs_regression(model_list, rainevents, watersheds, i, this_site)

        print '----- Watershed ',this_site['Basin_ID'],'i =',i,'-----'
        print ''
        print ' V_max =',V_max.round(2), ' S_min =',S_min.round(2), ' RMSE =', RMSE.round(2)
        print ' Q_s = [(W_eff - (' ,S_min.round(2),'))^2] / [', V_max.round(2), ' + W_eff - (' ,S_min.round(2),')]'
        print ''
    else:
        print '----- Watershed ',this_site['Basin_ID'],'i =',i,'-----'
        print ''
        print 'Not enough rain events'
        print ''
    watersheds['Evaporation included?'][model_list.index[i]] = 'yes'

print ' Bins =',(len(Bins)-1), ' Data Points =',watersheds['No of data points'][model_list.index[i]]
watersheds['No of bins'][model_list.index[i]] = len(Bins)-1
RainData[i] = 'Rain Data %s.csv' % int(model_list['Basin_ID'][model_list.index[i]])

```

```

Summary_of_rain_events[i] = 'Summary of Rain Events %s.csv' % int(model_list["Basin_ID"][model_list.index[i]])

# Save data to specific files
if watersheds["No of bins"][model_list.index[i]] > 2:
    shutil.move(RainData[i],raindata_directory)
    shutil.move(Summary_of_rain_events[i],summary_directory)

if NSE >= 0.8 and NSE <= 1.0:
    Graph_08to10[i] = 'Graph %s.png' % int(model_list["Basin_ID"][model_list.index[i]])
    shutil.move(Graph_08to10[i],G_NSE_08to10)

if NSE >= 0.6 and NSE < 0.8:
    Graph_06to08[i] = 'Graph %s.png' % int(model_list["Basin_ID"][model_list.index[i]])
    shutil.move(Graph_06to08[i],G_NSE_06to08)

if NSE >= 0.4 and NSE < 0.6:
    Graph_04to06[i] = 'Graph %s.png' % int(model_list["Basin_ID"][model_list.index[i]])
    shutil.move(Graph_04to06[i],G_NSE_04to06)

if NSE >= 0.2 and NSE < 0.4:
    Graph_02to04[i] = 'Graph %s.png' % int(model_list["Basin_ID"][model_list.index[i]])
    shutil.move(Graph_02to04[i],G_NSE_02to04)

if NSE >= 0.0 and NSE < 0.2:
    Graph_00to02[i] = 'Graph %s.png' % int(model_list["Basin_ID"][model_list.index[i]])
    shutil.move(Graph_00to02[i],G_NSE_00to02)

if NSE < 0.0:
    Graph_negative[i] = 'Graph %s.png' % int(model_list["Basin_ID"][model_list.index[i]])
    shutil.move(Graph_negative[i],G_NSE_negative)

if Weff_NSE >= 0.8 and Weff_NSE <= 1.0:
    Weff_Graph_08to10[i] = 'Weff Graph %s.png' % int(model_list["Basin_ID"][model_list.index[i]])
    shutil.move(Weff_Graph_08to10[i],NSE_08to10)

if Weff_NSE >= 0.6 and Weff_NSE < 0.8:
    Weff_Graph_06to08[i] = 'Weff Graph %s.png' % int(model_list["Basin_ID"][model_list.index[i]])
    shutil.move(Weff_Graph_06to08[i],NSE_06to08)

if Weff_NSE >= 0.4 and Weff_NSE < 0.6:
    Weff_Graph_04to06[i] = 'Weff Graph %s.png' % int(model_list["Basin_ID"][model_list.index[i]])
    shutil.move(Weff_Graph_04to06[i],NSE_04to06)

if Weff_NSE >= 0.2 and Weff_NSE < 0.4:
    Weff_Graph_02to04[i] = 'Weff Graph %s.png' % int(model_list["Basin_ID"][model_list.index[i]])
    shutil.move(Weff_Graph_02to04[i],NSE_02to04)

if Weff_NSE >= 0.0 and Weff_NSE < 0.2:

```



```

Weff_Graph_00to02[i] = 'Weff Graph %s.png' % int(model_list['Basin_ID'][model_list.index[i]])
shutil.move(Weff_Graph_00to02[i],NSE_00to02)

if Weff_NSE < 0.0:
    Weff_Graph_negative[i] = 'Weff Graph %s.png' % int(model_list['Basin_ID'][model_list.index[i]])
    shutil.move(Weff_Graph_negative[i],NSE_negative)

os.chdir(maindirectory) # Change the directory
retval = os.getcwd() # Check current working directory
print "Directory changed successfully %s" % retval
watersheds.to_csv('watersheds.csv')

```

Code for - fa_getdata

```
import pandas
import os.path as path

def get_timeseries_data(ID_number_str, output_type='pandas', directory=None):
    """
    Load daily timeseries data from a csv data file.

    *ID_number_str*
    A string giving the ID number of the csv, e.g. ``'01048000'``

    *directory*
    Directory of the data. Defaults to current directory

    """
    if type(ID_number_str) is not str:
        raise Exception("ID_number_str must be a string, like '01048000'")
    if not directory:
        directory = path.curdir
    file_name = path.join(directory, str(ID_number_str) + '.txt')
    if not path.exists(file_name):
        raise Exception("I can't find the file! You asked for " + ID_number_str)
    with open(file_name) as data_file:
        ts = pandas.read_csv(data_file, na_values=['-99.0000'], delim_whitespace=True, parse_dates={'date': ['YR', 'Mnth', 'DY']}, index_col='date')
    if output_type=='pandas':
        return ts
    else:
        raise Exception("Not a valid option for 'output_type'")
```

Code for - fb_dQ_dt_AND_avg_discharge

```
import pandas as pd
import numpy as np

def dQ_dt_AND_avg_discharge(data, model_list, watersheds, i):

    discharge = data['OBS_RUN']
    dt = 1. # daily time series

    # CALCULATE AVERAGE DISCHARGE (Q1+Q2)/2 AND -dQ_dt = (Q1-Q2)/dt
    avg_discharge = pd.Series((discharge[:-1].values+discharge[1:].values)/2, index = range(len(discharge)-1))
    dQ_dt = pd.Series((discharge[:-1].values-discharge[1:].values)/dt, index = range(len(discharge)-1))

    # FILTER: Use only data where discharge is 10xs greater than PET and no precipitation
    low_PET = (data['OBS_RUN'] > (10*data['PET'])).values
    low_PET = np.logical_and(low_PET[:-1], low_PET[1:])
    no_rain = (data['PRCP'] == 0).values
    no_rain = np.logical_and(no_rain[:-1], no_rain[1:])
    avg_discharge = avg_discharge[np.logical_and(low_PET, no_rain)]
    dQ_dt = dQ_dt[np.logical_and(low_PET, no_rain)]

    # FILTER: drop values with zero average discharge because the log is -inf
    azero = np.nonzero(avg_discharge == 0)[0]
    azero = avg_discharge.index[azero]
    azero = np.append(azero, dQ_dt.index[np.isnan(dQ_dt)])
    avg_discharge[azero] = float('nan')
    avg_discharge = avg_discharge.dropna()
    dQ_dt[azero] = float('nan')
    dQ_dt = dQ_dt.dropna()

    watersheds['No of data points'][model_list.index[i]] = len(avg_discharge)

    return avg_discharge, dQ_dt
```

Code for – fc_Binning

```
import pandas as pd
import numpy as np

def Binning(model_list, avg_discharge, dQ_dt, watersheds, i):

    Bin = (max(np.log(avg_discharge)) - min(np.log(avg_discharge))) / 100
    Bins = np.arange(min(np.log(avg_discharge)), max(np.log(avg_discharge)+Bin), Bin)

    # Create pandas data frame
    pd_avg_discharge = pd.DataFrame(avg_discharge, columns=['Ave Discharge'])
    pd_avg_discharge_log = pd.DataFrame(np.log(avg_discharge), columns=['Ave Discharge Log'])
    pd_dQ_dt = pd.DataFrame(dQ_dt, columns=['Ave -dQ_dt'])
    ave_discharge = pd_avg_discharge_log.join(pd_avg_discharge)
    recession = ave_discharge.join(pd_dQ_dt)

    ###-----GROUP data by Bin and CALC mean, std dev, and std error of Bin-----

    # CALC number of data points in each Bin
    length_bin = recession.groupby(pd.cut(recession['Ave Discharge Log'], Bins)).count()

    # GROUP BINS TOGETHER until there are more than two data points in each Bin
    finished = False
    while not finished:
        too_few = np.nonzero(length_bin['Ave Discharge'] < 3)[0]
        if len(too_few) > 0:
            if len(too_few) == 1:
                Bins=np.delete(Bins, too_few[0])
                finished = True
            else:
                Bins=np.delete(Bins, too_few[0]+1)
                length_bin = recession.groupby(pd.cut(recession['Ave Discharge Log'], Bins)).count()
        else:
            finished = True

    # CALC mean, std dev, and std error of Bin
    mean_bin = recession[['Ave Discharge', 'Ave -dQ_dt']].groupby(pd.cut(recession['Ave Discharge Log'], Bins)).mean()
    std_bin = recession[['Ave Discharge', 'Ave -dQ_dt']].groupby(pd.cut(recession['Ave Discharge Log'], Bins)).std()
    stderr_dQ_dt = (std_bin['Ave -dQ_dt'] / np.sqrt(length_bin['Ave -dQ_dt']))
    finished = False

    # GROUP BINS TOGETHER until the std error is less than half the mean of -dQ_dt
    while not finished:
        too_much_err = np.nonzero(stderr_dQ_dt > mean_bin['Ave -dQ_dt']*0.5)[0]
        if len(too_much_err) > 0:
            if len(too_much_err) == 1:
```

```

    Bins=np.delete(Bins, too_much_err[0])
    finished = True
else:
    Bins=np.delete(Bins, too_much_err[0]+1)
    # CALC mean, std dev, and std error of Bin
    length_bin = recession.groupby(pd.cut(recession['Ave Discharge Log'], Bins)).count()
    mean_bin = recession.groupby(pd.cut(recession['Ave Discharge Log'], Bins)).mean()
    std_bin = recession.groupby(pd.cut(recession['Ave Discharge Log'], Bins)).std()
    std_bin['Ave -dQ_dt'].fillna(0, inplace=True)
    stderr_dQ_dt = (std_bin['Ave -dQ_dt'] / np.sqrt(length_bin['Ave -dQ_dt']))
else:
    finished = True

return Bins, mean_bin, stderr_dQ_dt

```

Code for – fd_sensitivity_regression

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import fmin
from scipy.special import erf

def sensitivity_regression(model_list, events, Bins, mean_bin, stderr_dQ_dt, avg_discharge, dQ_dt, this_site, watersheds,i):

    w = np.log(1/stderr_dQ_dt) # weight
    ln_avg_Q = np.log(mean_bin['Ave Discharge'])
    ln_avg_dQdt = np.log(mean_bin['Ave -dQ_dt'])

    mean_curve = np.mean(np.log(mean_bin['Ave Discharge']))
    std_curve = np.std(np.log(mean_bin['Ave Discharge']))

    guess = np.array([ min(ln_avg_Q)-np.log(2), 1, 1])

    def objective(coeff):
        a = np.exp(coeff[0])
        bL = np.exp(coeff[1])
        bU = np.exp(coeff[2])
        # This varies b as a sigmoidal function between bL and bU
        b = bL + (bU - bL) * (1 + erf((ln_avg_Q-mean_curve)/(np.sqrt(2)*std_curve)))/2
        ln_dQdt = b * (ln_avg_Q - mean_curve) - np.log(a)
        RMSE = (np.sum(w*((ln_dQdt - ln_avg_dQdt) ** 2)) / len(ln_avg_dQdt)) ** 0.5
        return RMSE

    coeff = fmin(objective, guess) # minimize the objective function

    def dQdt_eq(Q, coeff):
        a = np.exp(coeff[0])
        bL = np.exp(coeff[1])
        bU = np.exp(coeff[2])
        b = bL + (bU - bL) * (1 + erf((Q-mean_curve)/(np.sqrt(2)*std_curve)))/2
        return b * (Q - mean_curve) - np.log(a)

    a = np.exp(coeff[0])
    bL = np.exp(coeff[1])
    bU = np.exp(coeff[2])

    watersheds['a'][model_list.index[i]] = a
    watersheds['bL'][model_list.index[i]] = bL
    watersheds['bU'][model_list.index[i]] = bU

    ln_dQdt = dQdt_eq(ln_avg_Q, coeff)
    NSE = 1-np.sum((ln_avg_dQdt-ln_dQdt)**2)/np.sum((ln_avg_dQdt-np.mean(ln_avg_dQdt))**2)
```

```

ln_Q_pred = (np.linspace(min(np.log(avg_discharge)), max(np.log(avg_discharge)), 100))
ln_dQdt_pred = dQdt_eq(ln_Q_pred, coeff)
dQ_dt_pred = np.exp(ln_dQdt_pred)
Q_pred = np.exp(ln_Q_pred)

watersheds['g(Q) NSE'][model_list.index[i]] = NSE

# PLOT DATA: Plot data point; Bins; weighted, ordinary, and poly regression
fig = plt.figure(1)
fig.clf()
ax = fig.add_subplot(111, axisbg='w')
data_pts = ax.plot(avg_discharge, dQ_dt, 'o', c='blue', markeredgcolor='none', alpha=0.2)
bins = ax.plot(mean_bin['Ave Discharge'], mean_bin['Ave -dQ_dt'], 'o', c='red', markeredgcolor='none')
sens = ax.plot(Q_pred, dQ_dt_pred, c='green', lw=1., alpha=1.0)
plt.legend((sens, data_pts, bins), ('Sensitivity Regression - NSE: %s'%round(NSE,2), 'Data Points', 'Bins'), loc='upper left', fontsize=8)
plt.title(this_site['Basin_ID'], fontsize = 16)
plt.xlabel('Discharge (Q, mm/day)', fontsize = 14)
plt.ylabel('-dQ/dt (mm/$\mathregular{day}^{\{2\}}$)', fontsize = 14)
ax.set_yscale('log', fontsize = 14)
ax.set_xscale('log', fontsize = 14)
ax.spines["top"].set_visible(False)
ax.spines["right"].set_visible(False)
ax.get_xaxis().tick_bottom()
ax.get_yaxis().tick_left()
plt.xlim(min(avg_discharge)-0.1*min(avg_discharge), max(avg_discharge)+5)
plt.ylim(min(dQ_dt[dQ_dt>0])-0.1*min(dQ_dt[dQ_dt>0]), max(dQ_dt)+5)
plt.savefig('Graph %d' % (this_site['Basin_ID']))

return coeff, NSE, dQdt_eq

```

Code for – fe_find_event_storage

```
import pandas as pd
import numpy as np
from scipy.integrate import cumtrapz

def find_event_storage(dQdt_eq, model_list, coeff, events, data, this_site, watersheds, i):

    place = np.zeros(len(data))
    d = {'PRCP': place, 'Store mm': place, 'Q_base': place, 'Q_storm': place, 'Q_total': place}
    raindata = pd.DataFrame(data=d, index=data.index)

    g = lambda Q: np.exp(dQdt_eq(np.log(Q), coeff))/Q
    if events['OBS_RUN'].min() <= 0:
        minimum = np.log(0.01)
    else:
        minimum = np.log(events['OBS_RUN'].min())
    x = np.exp(np.linspace(minimum, np.log(events['OBS_RUN'].max()), 1000, endpoint=True))
    y = 1./g(x)
    S = cumtrapz(y, x, initial=0) # area under curve
    S_t = np.interp(events['OBS_RUN'], x, S) # Find the S associated with observed flow thru interpolation
    events['Store mm'] = S_t - np.mean(S_t) # Storage relative to the mean of all the storages

    # LOCATE RANGES OF DAYS WITH NO PRECIPITATION (event precipitation will occur between ranges)
    a = events['PRCP']
    def zero_runs(a):
        # Create an array that is 1 where a is 0, and pad each end with an extra 0.
        iszero = np.concatenate(([0], np.equal(a, 0).view(np.int8), [0]))
        absdiff = np.abs(np.diff(iszero))
        # Runs start and end where absdiff is 1.
        ranges = np.where(absdiff == 1)[0].reshape(-1, 2)
        return ranges
    zeros = zero_runs(a)

    # CREATE PLACEHOLDERS for future array
    Si = np.zeros(len(zeros)) # initial storage
    Si_index = np.zeros(len(zeros))
    Sf = np.zeros(len(zeros)) # final storage
    W = np.zeros(len(zeros)) # total rainfall
    Q_s = np.zeros(len(zeros)) # storm flow
    W_eff = np.zeros(len(zeros))
    Storage = np.zeros(len(zeros))

    ###-----PERFORM ANALYSIS EVENT BY EVENT-----

    for m in range(len(zeros)-1): # for each range of zeros
```



```

# IF STATEMENT: Only analyze rain events with 3 or more days of NO rain BEFORE and AFTER the rain event
if (zeros[m][1] - zeros[m][0]) < 3 or np.isnan(events['Store mm'][zeros[m][0]:zeros[m][1]]).all() or (events['OBS_RUN'][zeros[m][0]:zeros[m][1]] < 0).any():
    events[zeros[m][0]:zeros[m][1]] = 0 # where theres < 5 days of no rain before event or nans before rain

elif (zeros[m+1][1] - zeros[m+1][0]) < 3 or np.isnan(events['Store mm'][zeros[m+1][0]:zeros[m+1][1]]).all() or (events['OBS_RUN'][zeros[m+1][0]:zeros[m+1][1]] < 0).any():
    events[zeros[m+1][0]:zeros[m+1][1]] = 0 # where theres < 5 days of no rain after event or nans after rain

elif (events['OBS_RUN'][zeros[m][0]:zeros[m+1][1]-1].values > events['OBS_RUN'][zeros[m][0]+1:zeros[m+1][1]].values).all():
    events[zeros[m][0]:zeros[m][1]] = 0 # where flow is always decreasing

elif (zeros[m][1] - zeros[m][0]) >= 3 and (zeros[m+1][1] - zeros[m+1][0]) >= 3: # where theres > 5 days before and after rain event

# ASSIGN rainevent to analyze, CALCULATE Si
rainevent = events[['PRCP', 'Store mm', 'OBS_RUN']][zeros[m][0]:zeros[m+1][1]]
Si[m] = events['Store mm'][zeros[m][1]-1:zeros[m][1]] #first day before it rains
Si_index[m] = zeros[m][1] - zeros[m][0] - 1

# CALCULATE lengths and volumne of rain
days_of_precip = len(rainevent[rainevent['PRCP']!=0]) # length of precipitation
total_length = days_of_precip + zeros[m][1] - zeros[m][0] + zeros[m+1][1] - zeros[m+1][0] # from beginning to end of event
length_so_far = days_of_precip + 1 + zeros[m+1][1] - zeros[m+1][0] # from Si to end of event
W[m] = np.sum(rainevent['PRCP'])

# CALCULATE BASEFLOW (use storage discharge equation to est - g(Q)) [[use lowest flow before storm]]
# Baseflow Calcs from Kirchner 2009 --> par 63
Q_base = np.zeros(length_so_far)
tau = np.zeros(length_so_far)
Q_base[1] = min(data['OBS_RUN'][zeros[m][1]-3:zeros[m][1]]) # lowest flow of 3 days before storm
tau[1] = 1./g(Q_base[1])
for j in range(int(length_so_far)-2):
    k = j+2
    j = j+1
    Q_base[k] = Q_base[j]*np.exp(-1/tau[j])
    tau[k] = 1./g(Q_base[k])
Q_base = np.nan_to_num(Q_base) # replace nans with 0

# CALCULATE STORM FLOW AND PRECIPITATION
Q_storm = np.zeros(length_so_far) # Q_storm = Q_observed - Q_base
Q_storm[1:] = rainevent['OBS_RUN'][int(Si_index[m])+1:int(total_length+Si_index[m])] - Q_base[1:]
Q_event = pd.DataFrame(data={'Q_storm': Q_storm, 'Q_base': Q_base}, index=rainevent.index[int(Si_index[m]):])
Q_event['Q_total'] = rainevent['OBS_RUN'][int(Si_index[m]):int(total_length+Si_index[m])]
Q_event['PRCP'] = rainevent['PRCP'][int(Si_index[m]):int(total_length+Si_index[m])]
Q_event['Store mm'] = rainevent['Store mm'][int(Si_index[m]):int(total_length+Si_index[m])]

# CALCULATE Sf
#initialize
for h in range((zeros[m+1][1] - zeros[m+1][0])-1):
    f = days_of_precip + 1 + h

```

```

if Q_event['Q_storm'][Q_event.index[f]] >= 0.1 * Q_event['Q_storm'][:f].sum():
    f=f
    if h == (zeros[m+1][1] - zeros[m+1][0])-2:
        Sf[m] = rainevent['Store mm'][rainevent.index[zeros[m][1]-zeros[m][0]-1 + f]]
        Sf_index = rainevent.index[zeros[m][1]-zeros[m][0] + f]
        break
    elif Q_event['Q_storm'][Q_event.index[f]] <= 0.1 * Q_event['Q_storm'][:f].sum():
        Sf[m] = rainevent['Store mm'][rainevent.index[zeros[m][1]-zeros[m][0]-1 + f]]
        Sf_index = rainevent.index[zeros[m][1]-zeros[m][0] + f]
        break

Q_event.ix[Sf_index:] = 0
Q_event = Q_event.loc[~Q_event.apply(lambda row: (row==0).all(), axis=1)]

# CALCULATE CUMULATIVE STORM FLOW, EFF PRECIP, AND STORAGE
Q_s[m] = Q_event['Q_storm'].sum()
W_eff[m] = (W[m] + Si[m])
Storage[m] = Sf[m] - Si[m]

raindata['PRCP'][Q_event.index[0]:Q_event.index[len(Q_event)-1]] = Q_event['PRCP']
raindata['Q_base'][Q_event.index[0]:Q_event.index[len(Q_event)-1]] = Q_event['Q_base']
raindata['Q_storm'][Q_event.index[0]:Q_event.index[len(Q_event)-1]] = Q_event['Q_storm']
raindata['Q_total'][Q_event.index[0]:Q_event.index[len(Q_event)-1]] = Q_event['Q_total']
raindata['Store mm'][Q_event.index[0]:Q_event.index[len(Q_event)-1]] = Q_event['Store mm']

# ORGANIZE THE VARIABLES INTO ONE ARRAY (delete events that didn't meet the 5 day criteria
# and storms that have less than 10 mm of precipitation)

rainevents = {'Si': Si, 'Sf': Sf, 'W': W, 'Effective inputs (Weff)': W_eff, 'Storm Flow Vol Obs (Q_s_obs)': Q_s, 'Vol in Storage': Storage}
rainevents = pd.DataFrame(data=rainevents)
rainevents = rainevents.loc[~rainevents.apply(lambda row: (row==0).all(), axis=1)]
rainevents = rainevents[rainevents['W']>10] #get rid of small rain events (<10mm)

raindata = raindata.loc[~raindata.apply(lambda row: (row==0).all(), axis=1)]
raindata.to_csv('Rain Data %d.csv' % (this_site['Basin_ID']))

watersheds['min flow'][model_list.index[i]] = minimum
watersheds['max flow'][model_list.index[i]] = np.log(events['OBS_RUN'].max())
watersheds['mean storage'][model_list.index[i]] = np.mean(S_t)

return rainevents

```

Code for – fg_least_square_Qs_regression

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import fmin
from scipy.optimize import fmin_cobyla

def least_square_Qs_regression(model_list, rainevents, watersheds, i, this_site):
    Q_s_obs = rainevents['Storm Flow Vol Obs (Q_s_obs)'].values
    W_eff = rainevents['Effective inputs (Weff)'].values
    rainevents['Storm Flow Vol Pred (Q_s_pred)'] = np.zeros(len(rainevents))

    def min_this(params): # Calculate Vmax and Smin and equation for Qs
        V_max = np.exp(params[0])
        S_min = params[1]
        Q_s_pred = np.where(W_eff > S_min, (W_eff - S_min)**2 / (V_max + W_eff - S_min), 0.)
        err = np.sqrt(np.mean(((Q_s_obs) - (Q_s_pred))**2))
        return err

    V_max_guess, S_min_guess = 0., rainevents['Si'].min() + (rainevents['Si'].min() - rainevents['Si'].mean())/2
    params = fmin(min_this, np.array([V_max_guess, S_min_guess]))

    if np.exp(params[0]) > params[1]:
        V_max = np.exp(params[0])
        S_min = params[1]
        watersheds['V_max < S_min?'][model_list.index[i]] = 'no'
    else:
        watersheds['V_max < S_min?'][model_list.index[i]] = 'yes'
    def constr1(params):
        np.exp(params[0]) - params[1]
    def constr2(params):
        return np.exp(params[0])

    V_max_guess, S_min_guess = 0., rainevents['Si'].min() + (rainevents['Si'].min() - rainevents['Si'].mean())/2
    params = fmin_cobyla(min_this, [V_max_guess, S_min_guess], [constr1, constr2], rhoend=1e-7, iprint=1)
    V_max = np.exp(params[0])
    S_min = params[1]

    Q_s_pred = np.where(W_eff > S_min, (W_eff - S_min)**2 / (V_max + W_eff - S_min), 0.)
    RMSE = np.sqrt(np.mean((Q_s_obs - Q_s_pred)**2))
    rainevents['Storm Flow Vol Pred (Q_s_pred)'] = Q_s_pred

    NSE = 1 - (np.sum((Q_s_obs - Q_s_pred)**2) / np.sum((Q_s_obs - np.mean(Q_s_obs))**2))

    watersheds['RMSE'][model_list.index[i]] = RMSE
```

```

watersheds["V_max"][model_list.index[i]] = V_max
watersheds["S_min"][model_list.index[i]] = S_min
watersheds["Weff NSE"][model_list.index[i]] = NSE

rainevents["Storm Flow Vol Pred (Q_s_pred)"] = Q_s_pred
for j in range(len(rainevents)):
    if rainevents["Storm Flow Vol Pred (Q_s_pred)"][rainevents.index[j]]==0:
        rainevents.ix[rainevents.index[j]]=0

rainevents = rainevents.loc[~rainevents.apply(lambda row: (row==0).all(), axis=1)]
rainevents.to_csv('Summary of Rain Events %d.csv' % (this_site["Basin_ID"]))

# PLOT FITTED EQUATION AND DATA
fig = plt.figure(2)
plt.clf()
fig.add_subplot(111, axisbg='w')
observed, = plt.plot(rainevents["Effective inputs (Weff)"], rainevents["Storm Flow Vol Obs (Q_s_obs)"], 'b.')
predicted, = plt.plot(rainevents["Effective inputs (Weff)"].order(), rainevents["Storm Flow Vol Pred (Q_s_pred)"].order(), 'r')
plt.legend((observed, predicted),("Weff Observed","Weff Predicted - NSE: %s"%round(NSE,2) ),loc='upper left',fontsize=8)
plt.title("Observed and predicted event runoff for %s"% (this_site["Basin_ID"]), fontsize = 16)
plt.xlabel("$W_{eff}$ [mm]", fontsize = 14)
plt.ylabel("$Q_s$ [mm]", fontsize = 14)
plt.savefig('Weff Graph %d' % (this_site["Basin_ID"]))

return V_max, S_min, RMSE, NSE

```

Code for – fh_dQ_dt_AND_evap_term

```
import pandas as pd
import numpy as np

def dQ_dt_AND_evap_term(model_list, data, watersheds, i):

    discharge = data['OBS_RUN']
    evap = data['ET']
    dt=1.

    # CALCULATE AVERAGE DISCHARGE (Q1+Q2)/2 AND -dQ_dt = (Q1-Q2)/dt
    avg_discharge = pd.Series((discharge[:-1].values+discharge[1:].values)/2, index = range(len(discharge)-1))
    avg_evap = pd.Series((evap[:-1].values+evap[1:].values)/2, index = range(len(evap)-1))
    E_Q_plus_one = avg_evap/avg_discharge + 1
    dQ_dt = pd.Series((discharge[:-1].values-discharge[1:].values)/dt, index = range(len(discharge)-1))
    dQ_dt_EQone = dQ_dt / E_Q_plus_one

    # FILTER: Use only data where discharge is 10xs greater than PET and no precipitation
    no_rain = (data['PRCP']==0).values
    no_rain = np.logical_and(no_rain[:-1],no_rain[1:])
    dQ_dt_EQone = dQ_dt_EQone[no_rain]
    avg_discharge = avg_discharge[no_rain]

    # FILTER: drop values with zero average discharge because the log is -inf
    azero = np.nonzero(avg_discharge==0)[0]
    azero = avg_discharge.index[azero]
    azero = np.append(azero,dQ_dt_EQone.index[np.isnan(dQ_dt_EQone)])
    avg_discharge[azero]=float('nan')
    avg_discharge=avg_discharge.dropna()
    dQ_dt_EQone[azero]=float('nan')
    dQ_dt_EQone=dQ_dt_EQone.dropna()

    watersheds['No of data points'][model_list.index[i]] = len(dQ_dt_EQone)

    return avg_discharge, dQ_dt_EQone
```

Curriculum Vitae

Holly M. Guest, E.I.

Birth date: June 7th, 1991

Johns Hopkins University, USA.

115 Inwood Trail
Madison, AL 35758, USA
Ph: +256-503-9316
Email: hollyguest7@gmail.com
Update on: April 2015

Research Experience

While receiving my undergraduate degree, I assisted graduate student Carmen Chosie in her experimental work involving the use of high rate settlers for construction site sediment basins. The experiments on the scale model proved that high rate settlers trapped sediment from the forebay, thus decreasing the turbidity of the outflow.

I also assisted Ph.D. student Mitchell Moore in water quality monitoring for Little Cahaba Creek in Birmingham, AL. This project correlates water quality, highway traffic, and the hydrological environment to determine the impact that the highways have on the nearby and intersecting streams. I have experience setting up a variety of experimental apparatuses, performing hydraulic measurements for turbidity and flow velocity, and conducting water quality tests such as TSS and total nitrogen.

While at Johns Hopkins University I was involved with putting together a thesis. The thesis combines two common methods used to analyze the hydrologic response of watersheds to improve our ability to develop a catchment classification system. The combined methods were applied to a large data set of watersheds and then the model's parameters were mapped spatially to identify trends.

Academic Training

I am graduating from Johns Hopkins University in May with a MS degree in Water Resource Engineering. I am an graduate from Auburn University with a Bachelor of Civil Engineering, graduating summa cum laude with a 3.81/4.00 GPA.

While at Auburn, I took the Groundwater Hydraulics class and the Open Channel Hydraulics class where I learned MODFLOW and HEC-RAS, respectively. I also took Urban Hydraulics where I learned how to design and manage urban water supply, sanitary sewer, storm water collection systems, and flood control works.

My senior capstone project entailed designing an alternative disinfectant for Madison, Georgia's water treatment plant. The plant currently uses chlorine gas and my team designed two alternatives: bulk sodium hypochlorite and on site hypochlorite generation. Once completed, we presented our design, cost estimate, and recommendation to our professor, consultant, and Madison's city manager.

I took six months off and volunteered for Engineering Ministries International where I took a trip to Colombia in February 2014 to aid in the design of the water distribution system and waste water system for a preschool.

While at Johns Hopkins, I focused in on hydrology taking related classes in climate and water treatment. I became heavily involved in my research project by learning Python and applying what I learned from my classes.

Academic/Professional Appointments

2014 Civil Engineering Intern, Engineering Ministries International

- Assist in leading a team of professionals to Colombia offering free design services to design a master plan of a school.
- Draft drawings in AutoCAD, write the project report, coordinate between design professionals when back in the United States

- Assist in design of civil portion of the project and present the final design to the owner
- 2013 Hydraulics Teaching Assistant, Auburn University.
- Responsible for leading students in laboratory exercises.
 - Grade homework assignments and lab reports.
- 2012 - 2013 Hydraulics Research Assistant, Auburn University.
- Design and setup experimental apparatuses.
 - Manage laboratory runs for high rate settlers and traditional sedimentation experiments.
 - Collect and analyze data from all laboratory exercises.
- 2011 - 2012 Civil Engineering Co-Op Student, Brasfield & Gorrie, LLC.
- Assumed Project Manager responsibilities as a general contractor for a facilities building.
 - Responsible for estimating various projects
 - Performed cost analysis on companies concrete work

Software Training

- MODFLOW
- HEC-RAS
- AutoCAD
- GIS
- Microsoft Office Professional package (Word, Excel, PowerPoint, Publisher, Access)
- On Screen Take-Off
- Windows and Mac based PC

Software Training

- Python
- R
- VisualBasic
- MATLAB

Honors, Awards and Recognitions

- Engineering Intern (2013)
- American Water Works Association Scholarship (2013)
- Contractor's License Fee Scholarship (2013)
- George L. McGlamery Endowed Scholarship (2013)
- University Scholarship (2009)

Memberships

- American Society of Civil Engineering (ASCE)
- Chi Epsilon (Civil Engineering Honor Society)
- Tau Beta Pi (Engineering Honor Society)
- Phi Kappa Phi (Honor Society - top 10% of class)
- National Society of Leadership and Success